ETL-0472

AD-A188 108

Contour-to-Grid Interpolation with Nonlinear Finite Elements: A Feasibility Study

Betty Mandel
Javier Bernal
Christoph Witzgall

U.S. Army Corps of Engineers
Engineer Topographic Laboratories
Fort Belvoir, VA 22060-5546

Center for Applied Mathematics
National Bureau of Standards
Gaithersburg, MD 20899

September 1987

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release ; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): ETL-0472	5. MONITORING ORGANIZATION REPORT NUMBER(S): ETL-0472
6a. NAME OF PERFORMING ORGANIZATION: U.S. Army Engineer Topographic Laboratories; 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION: Center for Applied Mathematics, National Bureau of Standards
6c. ADDRESS (City, State, and ZIP Code): Fort Belvoir, VA 22060-5546	7b. ADDRESS (City, State, and ZIP Code): Gaithersburg, MD 20899
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

Contour-to-Grid Interpolation with Nonlinear Finite Elements: A Feasibility Study

12. PERSONAL AUTHOR(S)

Betty Mandel, Christoph Witzgall, Javier Bernal

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM Jan 86 TO Sep 86	September 1987	71

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

cartography, Clough-Tocher element, contour line, Delaunay diagram, digitized data, finite element method, line generalization, surface interpolation, synthetic surface, terrain modeling, Thiessen diagram, tolerance band, triangulation, Voronoi diagram.

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The U.S. Army Engineer Topographic Laboratories and the National Bureau of Standards investigated the computational feasibility of developing a contour-to-grid algorithm by: (1) sampling a large set of digitized cartographic contour data with a tolerance band technique, (2) its subsequent triangulation with a Voronoi (Delaunay, Thiessen) method and (3) the construction of an interpolating smooth synthetic surface from which grids can be generated at any given interval. This report presents a discussion on the results obtained.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
E. James Books	202-355-3039	CEETL-IM-T

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE

PREFACE

Our interest in this area was stimulated by Dr. Stephen Grotzinger and Mr. Douglas Caldwell. We are grateful to Mr. Arthur Noma of the Defense Mapping Agency Hydrographic/Topographic Center (DMAHTC) for providing the data for these demonstration efforts, and for his continuing interest and support. We appreciate Mr. Stephen Rogers' assistance in the preparation of plots needed to represent and update the data. Finally, we thank Mr. Steve Hasenfus, Mr. Larry Cook and Mr. Barry Holecheck for being there whenever an "emergency" occurred.

CONTENTS

1. INTRODUCTION

This report concludes the second phase of a three-phase collaborative effort between the U.S. Army Engineer Topographic Laboratories (ETL) and the National Bureau of Standards (NBS). This phase is the object of Interagency Agreement E8786K041 covering the time period from January to September 1986. The first phase of this effort took place from May to December 1985 under Interagency Agreement E8735K137. The third phase of this effort is in the planning stage.

The main objective of this effort is to develop and implement a "contour-to-grid" algorithm, that is, an algorithm capable of converting digitized contour data into Digital Terrain Elevation Data (DTED). The approach is to create a synthetic terrain surface based on digitized contour information and then to calculate any grid from that surface. Five tasks are to be performed:

1) Edit a digitized contour data set (input);
2) Thin (reduce, sample) the input data;
3) Triangulate the selected points;
4) Generate a smooth synthetic surface; and
5) Generate the desired grid.

The first phase, which was completed in December 1985, addressed the first three tasks. A description of the work done and the results obtained appear in Witzgall, Bernal and Mandel (85). The second phase, to which this report refers, elaborates on the previous results and includes work in all five areas. The effort demonstrates the feasibility of performing such tasks for large data sets, such as a Digital Graphic Recorder (DGR) data tape of roughly 500,000 points.

Details about the data structure used, the problems encountered, and the necessary transformations will be discussed in the second section of the report.

In the third section, we will describe a tolerance band algorithm for thinning the data set to a size suitable for triangulation and subsequent interpolation. This tolerance band algorithm is a one-pass version of the method by Reumann and Witkam (74) in analogy to techniques proposed by Williams (78,81). It was used to create sets of sample contour points, ranging from roughly 40,000 to 70,000 points. An algorithm previously developed by NBS was then employed to determine the Voronoi triangulations of these sets of points in a plane as illustrated in Figure 1. This process required, however, the development of a decomposition algorithm for data sets consisting of more than 50,000 points. This and other necessary adaptations are discussed in the fourth section of this report. Section Five will include a detailed discussion of the methodology employed to generate the surface, including the algorithm developed to produce rectangular grids for any given unit sizes. The testing procedure, along with a description of the structural layout used while performing the different tasks appears in the sixth section. This section will also contain a discussion of additional issues that are expected to be relevant to further work in this area.

Plots were obtained for the contour lines in their original form as well as in generalized form corresponding to 40,000 and 70,000 point samples using a Gerber Plotter. The Voronoi triangulation of a 40,000 point sample was also plotted. The surface generation software which was prepared as part of this effort, was applied to a a 40,000 point sample. The resulting surface was tested using the original data, independently of the sample data set.

All computations were initially carried out on a VAX 11/750 system and later (July 1986) transported to a VAX 11/780 system at ETL. They were found to be in the range of extensive, but routine computations of the kind that can be expected as part of the normal load of such systems.

Triangulation based surface modelling has been considered for some time, e.g. Peucker and Douglas (75). For other related work see Davis, Downing, and Zoraster (82), and Grotzinger, Danielson, Caldwell, and Mandel (84).

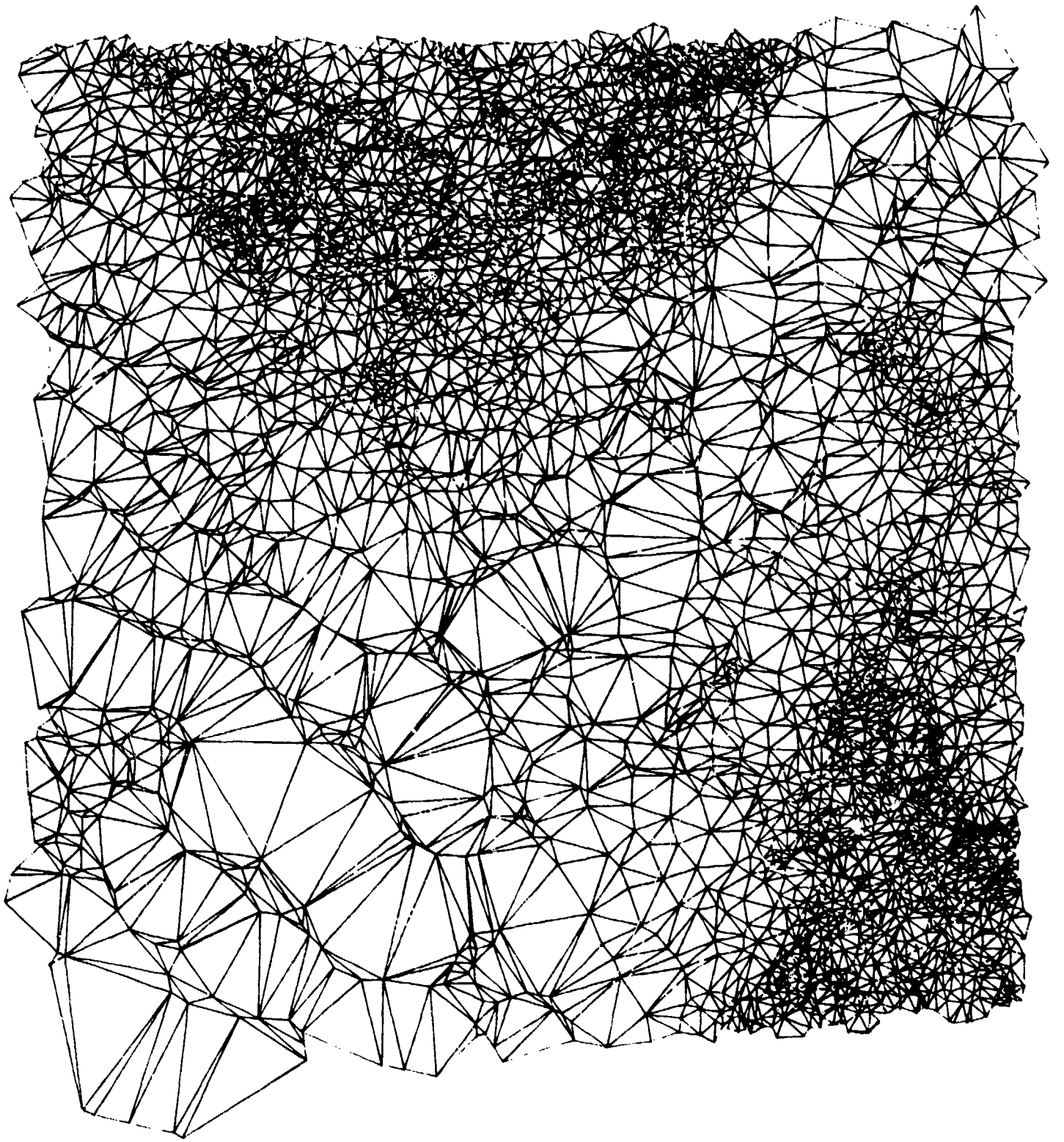

Figure 1. Portion of a Computer-Generated Plot of the Voronoi Triangulation of Selected Contour Points for the Mustang Mountain Area.

2. DIGITAL DATA STRUCTURE

The first task performed in this effort consisted of obtaining and editing a digital contour data set making it compatible with the software developed for the other tasks. All the necessary computations were performed originally in a VAX 11/750 system and then transported to a VAX 11/780 with a VMS operating system. Details of the modifications made to the particular data set used follow.

2.1 DIGITAL GRAPHIC RECORDER DATA

The process of digitizing cartographic information has been used for various purposes over the past decades. In particular, the Digital Graphic Recorder (DGR) has been employed to trace the lines of a map in order to generate sequences of coordinate pairs representing these lines. This digital data set is then stored on magnetic tape and is available in this form for computer processing. Specifically, we are interested in the problem of determining a computer internal terrain surface representation from this information.

An example of a data tape generated by a DGR was made available by the Defense Mapping Agency Hydrographic/Topographic Center (DMAHTC) together with a graphic plot derived from this information. The information had been generated from a 1:24,000 map of the Mustang Mountain area in Fort Huachuca, Arizona, -- an area which exhibits a large range of elevations and slopes. The digitizer resolution used, 0.01 inches, corresponds to a horizontal distance of 20 feet.

Contour related data were extracted from this tape and the information was reformatted and installed in a VAX system. Two basic problems were encountered in this effort concerning the format of the data and the size of the files. First of all, the records in a DGR magnetic tape contain 192 CDC-1700 words. Each word contains 18 bits of which only the high order 16 bits are used. We were to install this data set in a VAX with a VMS operating system. The words

in this system are 32 bits long and the bytes are interchanged in each word. To solve this problem a program called DGR2TAPE was written based on an example provided by DMAHTC (85). This program interchanges the high order with the low order bytes in each word and then it regroups the bits into 16-bit words by extracting the high order 16 bits from each 18-bit group.

The DGR data tape used included digitized contours, ridges, drains and neatlines. In the original file layout (DMAHTC, 85) that follows, segments of such traced lines are referred to as "scan lines":

FIRST RECORD: HEADER

WORD	CONTENTS
1- 5	Sheet Number
6-10	$$$$$
11-20	Operator's Initials
21-30	Date (07/16/84)
31-40	Scale (1:24000)
41-44	Series
45-50	$$$$$$
51	Edition
52-60	$$$$$$$$
61-68	Registration Marks in Binary (in 0.01 inches)
69-192	Comments

DATA RECORDS

WORD	CONTENTS
1	Scan Line Flag
2	Scan Line Number (I.D.)
3	Type of Data
4	X1
5	Y1 (in 0.01 inches)
6	Z1
*	*
*	*
*	*
3n+1	Xn
3n+2	Yn
3n+3	Zn

For each line or line segment that is digitized, the following information is recorded: a "Scan Line Starting Flag" (octal 200000) that indicates a new segment; a "Scan Line Number" that identifies the segment; a "Data Type Index" that classifies the segments as contours, neatlines, and ridges or drains; and one or more data points as illustrated in the Data Records Layout above. Each record may contain one or more traced lines, or depending on the size of the segment, it may contain only a portion of the segment. Note that a segment is any portion of a traced line, covering possibly the whole line, but more often dividing the traced line into several non-contiguous segments.

The data file consisted of 9023 blocks of 192 words each. Special constraints on the amount of disk space available and data accessibility for the type of manipulation intended guided the design of the new file layout. It was necessary to eliminate all the data not useful at this point of our investigation. The first record or header record was eliminated and from the data records only the contour line segments were kept. The resulting data format is described below.

DATA RECORDS

WORD	CONTENTS
1	Segment Flag
2	Segment Number (I.D.)
3	Z-Value (Elevation of Contour Line)
4	X1
5	Y1
*	*
*	*
*	*
2n+2	Xn
2n+3	Yn

2.2 EDITING THE INPUT DATA

An examination of the data showed that the contour lines extended beyond the neatlines. Leaving the contours unedited caused some undesirable triangles to appear in the triangulation. These triangles are very long and narrow and can be the source of large errors when a surface is generated. They are caused by the fact that the VORONOI triangulation produces a convex hull. A program called CLOSCONT was written to solve this problem and to join "neighboring" segments. This program edits the contours of the input data to include only information that lies within a rectangular area that can be chosen arbitrarily. The contour points falling beyond this boundary are deleted, so that the boundary includes the last point of each contour which was intersecting it. The resulting boundary contains points that are closer together, eliminating the undesired triangles. For the effort reported here, the boundaries were chosen so that the number of points to be deleted was kept to a minimum. CLOSCONT then searches the digital contour segments that remain for adjoining segments (neighbors) and joins them.

The nature of the digitized data is such that all points are in a grid with a point separation of 0.01 inches. Thus, neighboring points in a line are separated by either a vertical space, a horizontal space, or a diagonal, with a maximum separation between points of 0.01414 inches. An illustration of a digitized line appears in Figure 2. Integer numbers are used to identify each grid point although the distance between points is given in hundredths of an inch. Therefore, there is a finite number of points representing a line. It was also observed that there are suppressed contours in steep areas where the contour density is high. In these steep areas a small change in the horizontal location of a point has greater impact on its elevation computation and the resolution of the digitizer becomes more important. The effects caused by this fact are discussed in Section 6.1. Recall that the resolution of the digitizer, 0.01 inches, corresponds to a horizontal ground distance of 20 feet.

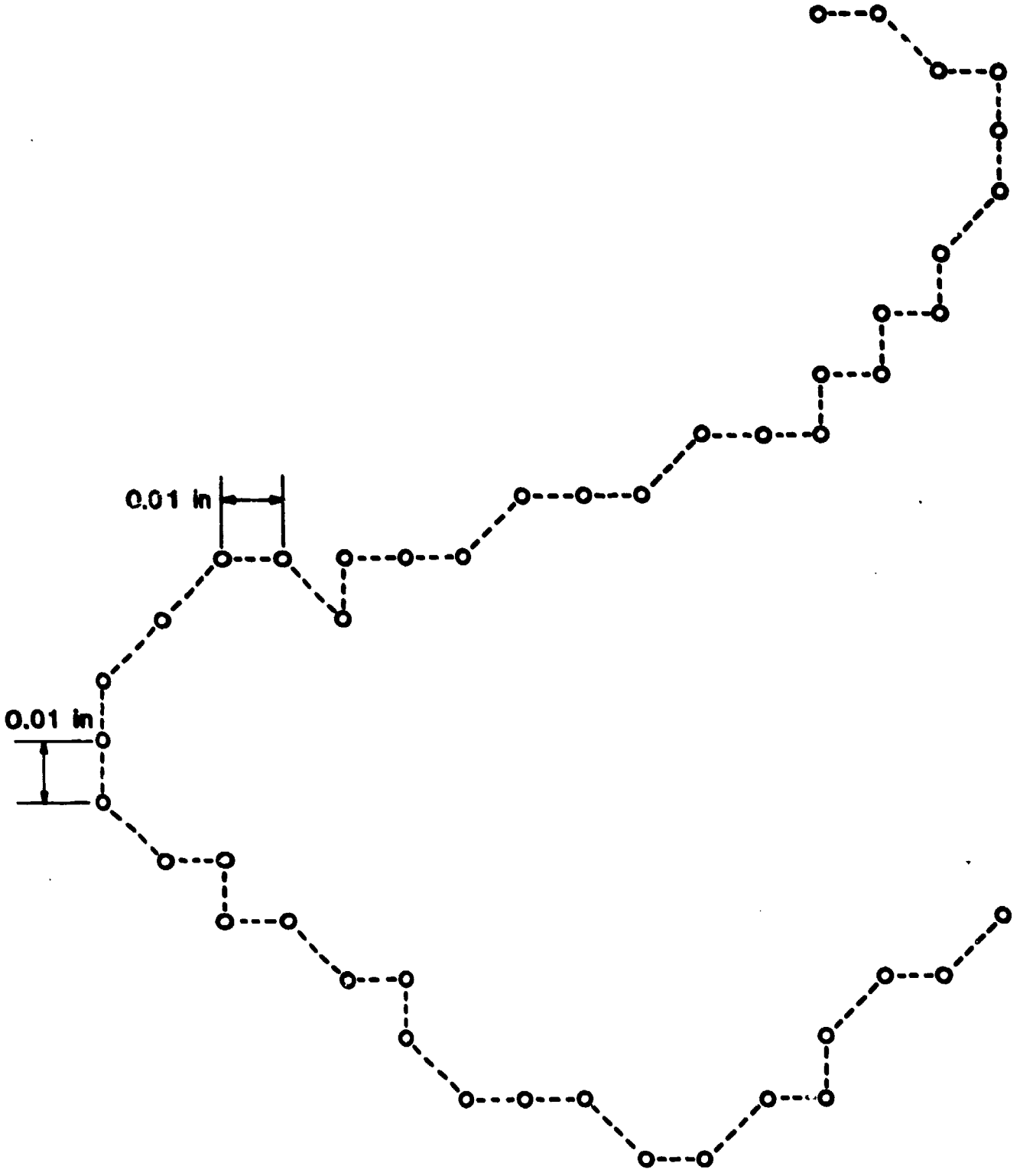

Figure 2. Portion of a Digitized Line

3. THINNING BY TOLERANCE BAND ALGORITHMS

For the purpose of reducing or thinning digitized line data, the second task of this effort, we selected a tolerance band method. This approach permits the linking of the sampling process to the accuracy of the approximation, selecting more points in areas of high curvature than in those of lower ones. We also calculate the contour tangents at most selected points except at the ends of segments. A program called THIN that performs all these computations was developed.

3.1 DEFINITIONS AND OVERVIEW

In general, the task of tolerance band methods is to select a suitable sample of points from an initial line . The points in the selected sample are frequently called "critical points". Connecting successive critical points by straight lines leads to a "generalized line" often desired for cartographic purposes (Figure 3).

The critical points are selected in such a fashion that the line points between two successive critical points are contained in a rectangular strip, the "tolerance band", whose bandwidth is determined from a given "tolerance" ε. The deviation of the initial line from the general line can thus be bounded in terms of ε.

Several varieties of tolerance band methods have been proposed by Lang (69), Douglas and Poiker (73), Reumann and Witkam (74) and Williams (78,81). For details the reader is asked to consult the comprehensive survey by Zoraster, Davis, and Hugus (84). For this project, we designed and used a one-pass algorithm for the method of Reumann and Witkam (74) in the spirit of Williams (78). It will be described below.

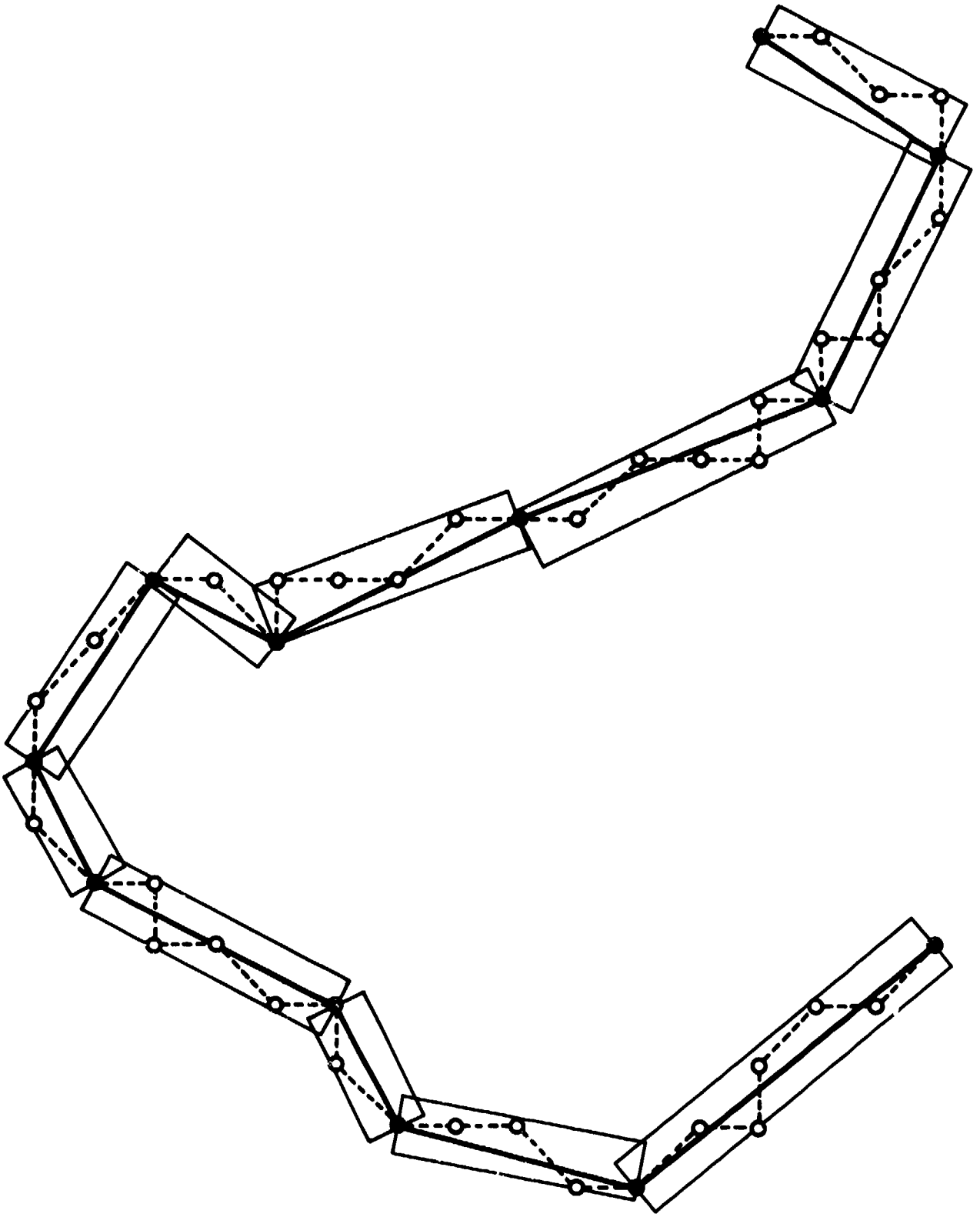

Figure 3. Covering of an Initial Line by Tolerance Bands . Solid Dots Represent Critical Points. Their Heavy Straight-Line Connections Define the Generalized Line.

3.2 A ONE-PASS VERSION OF THE REUMANN AND WITKAM METHOD

By

$$(3.2.1) \qquad \{ P_i = (x_i, y_i) \}, \; i=1,\ldots, n, \quad n > 1,$$

we denote the sequence of points in the initial line. In what follows, we will describe an algorithm that implements the method of Reumann and Witkam (74). Inherent in this method is the convention that the first initial point P_1 and the last initial point P_n are automatically selected as critical points. In addition we impose the condition that there should be no "doubling back" by the initial points within a tolerance band, that is, that

(3.2.2) distances from the first (critical) point to subsequent points i the tolerance band do not decrease when considered in sequence.

Furthermore we require that

(3.2.3) two critical points have at least distance ε from each other, unless they represent the first and last initial points.

With these conventions, the algorithm proceeds as follows. The point P_1 is automatically selected as the first critical point. As to subsequent points P_i, we consider the following conditions:

(3.2.4) (i) $i < n$
(ii) distance $(P_1, P_i) < \varepsilon$
(iii) distance $(P_1, P_i) \leq$ distance (P_1, P_{i+1})
(iv) there exists a tolerance band of the second kind anchored at P_1 which contains the points P_k, $1 \leq k \leq i+1$.

The point P_i is then noncritical if, in terms of the conditions (3.2.4),

(3.2.5) (i) and ((ii) or ((iii) and (iv))).

The algorithm checks P_2, P_3,... successively until a first critical point P_j is encountered. If $j<n$, that critical point will play the role of P_1, that is, it will anchor the search for a third critical point, and so on. In Figure 4 we give a pseudo-code description of the algorithm.

The major part of the algorithm concerns the verification of the tolerance band condition. Return to the two points P_1 and P_2, and assume that $n>2$ and that P_2 satisfies the ϵ-separation condition (3.2.4.ii). Then there exist unique left-most and right-most tolerance bands of width 2 anchored at P_1 which also contain P_2 (Figure 5). The flanks of these tolerance bands are tangent to the ϵ-circle around P_1. In particular, the right flank of the right-most tolerance band touches that circle at point R, whereas the left flank of the left-most tolerance band touches at point L. We are particularly interested in the area which lies to the right of the line from R to L, and which is bounded by rays extending from the points R to L in the direction of the right-most and left-most flanks, respectively (Figure 6). If the point P_3 satisfies the no-doubling-back condition (3.2.4.iii), then the subsequent condition(3.2.4.iv) holds clearly if and only if P_3 lies in this area.

In general, suppose that P_j is the last critical point to have been determined, and that P_i, $i<n$, as well as the points between it and P_j, are contained in at least one suitable tolerance band. Then, there are left-most and right-most tolerance bands -- in extreme cases they may coincide -- which contain those points also. Suppose further that P_{i+1} is not closer to P_j than P_i is. Then, there exists a tolerance band containing P_{i+1}, and all previous points back to P_j, if and only if P_{i+1} lies in the area indicated in Figure 6.

In this latter case, there exist again a left-most and a right-most tolerance band, each also covering the extended set $\{ P_j, P_{j+1}, \ldots, P_i, P_{i+1} \}$. These extremal positions are determined by the location of P_{i+1} with respect to the original extreme tolerance bands. To explain the situation, it is advantageous to introduce the notion of a "tolerance strip" associated with a tolerance band. By this we mean the infinite straight continuation of the

tolerance band in its direction. The tolerance strip thus is bounded by the origin-end of the tolerance band and the two infinite rays that extend the flanks of the band. We call those rays again the "flanks" of the tolerance strip. They are divided symmetrically by the "center line". In addition, we call the area described in Figure 6 the "area of flexibility" of P_i with respect to the anchor point P_j. This area of flexibility turns out to be the convex hull of the two tolerance strips associated with the right-most and left-most tolerance bands, respectively.

The point P_{i+1} now lies in (1) both tolerance strips, (2) the right-most tolerance strip only, (3) the left-most tolerance strip only, or (4) none. Two of these four cases are illustrated in Figures 7 and 8. In case (1), the original extreme tolerance strips remain unchanged. However, the tolerance bands cut from these strips are now, in general, longer since P_{i+1} is to lie on their destination ends. In case (1), only the right-most tolerance strip remains the same, as the direction of the left-most strip must now be changed to cover P_{i+1}. The smallest adjustment that will achieve this is a rotation of the left-most strip until its right flank meets P_{i+1}. This then characterizes the new position of the left-most tolerance strip. Case (3) is like case (2), except that the roles of the right-most and left-most tolerance strips are reversed. In case (4) finally, both tolerance strips need to be adjusted. The adjustment is such that the right flank of the new left-most strip and the left flank of the new right-most strip both intersect at P_{i+1}.

After a critical point P_j, $j<n$, has been established, and P_{j+1} has at least distance ε from P_j, the left-most and right-most tolerance strips are first established as if they were arising from case (4) above. This then establishes the area of flexibility. Each time a subsequent point is found to lie in this area, the tolerance strips are adjusted and the area of flexibility narrowed. If a subsequent point falls outside the area of flexibility, then its predecessor will be selected as a critical point. This also will be the case if doubling back occurs, or if there are no points left (compare the pseudo-code description in Figure 4).

```
define sets
   LS = LEFT-MOST TOLERANCE STRIP
   RS = RIGHT-MOST TOLERANCE STRIP
   AF = AREA OF FLEXIBILITY
define variables
   N    = NUMBER OF INITIAL POINTS (INPUT)
   P(I) = I-TH INITIAL POINT (INPUT)
   EPS  = TOLERANCE PARAMETER (INPUT)
   M    = NUMBER OF CRITICAL POINTS (OUTPUT)
if N<2 abort
K:=1; J:=1; C(1):=P(1)
while J<N do
   LS:=EMPTY; RS:=EMPTY; I:=J; CRITICAL POINT:=FALSE
   while not CRITICAL POINT do
      I:=I+1
      if |P(J)-P(I)| >EPS then
         if P(I) ∉ LS then ADJUST LS
         if P(I) ∉ RS then ADJUST RS
         DETERMINE AF
         if I=N or |P(J)-P(I)| > |P(J)-P(I+1)| or P(I+1) ∉ AF then
            CRITICAL POINT:=TRUE
            K:=K+1; J:=I; C(K):=P(J)
M:=K
```

Figure 4. Pseudo-Code Description of the One-Pass Reumann and Witkam Algorithm.

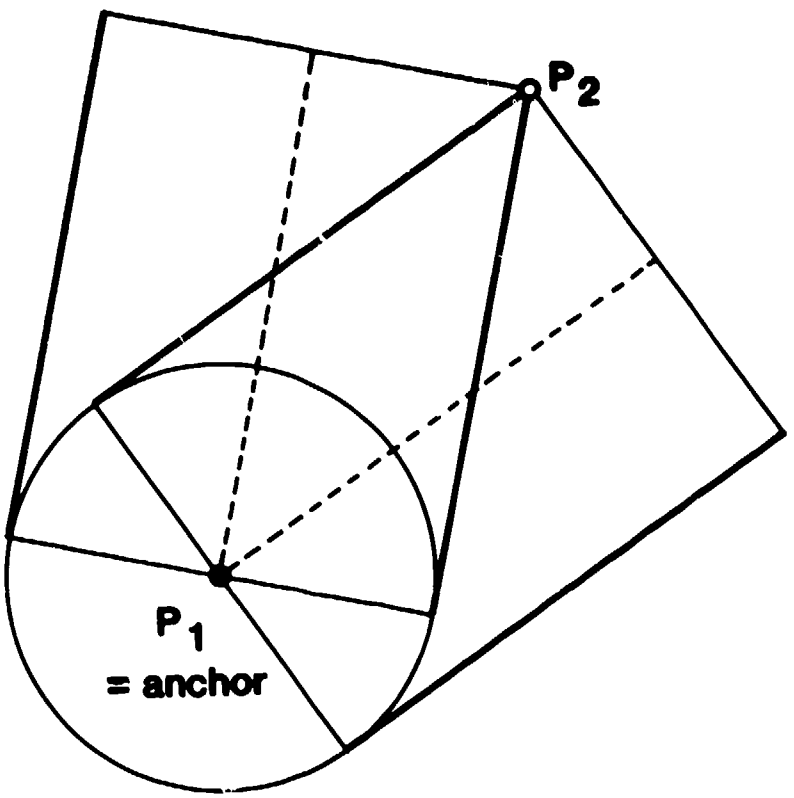

Figure 5. Left-Most and Right-Most Tolerance Bands Anchored at P_1 and Containing P_2.

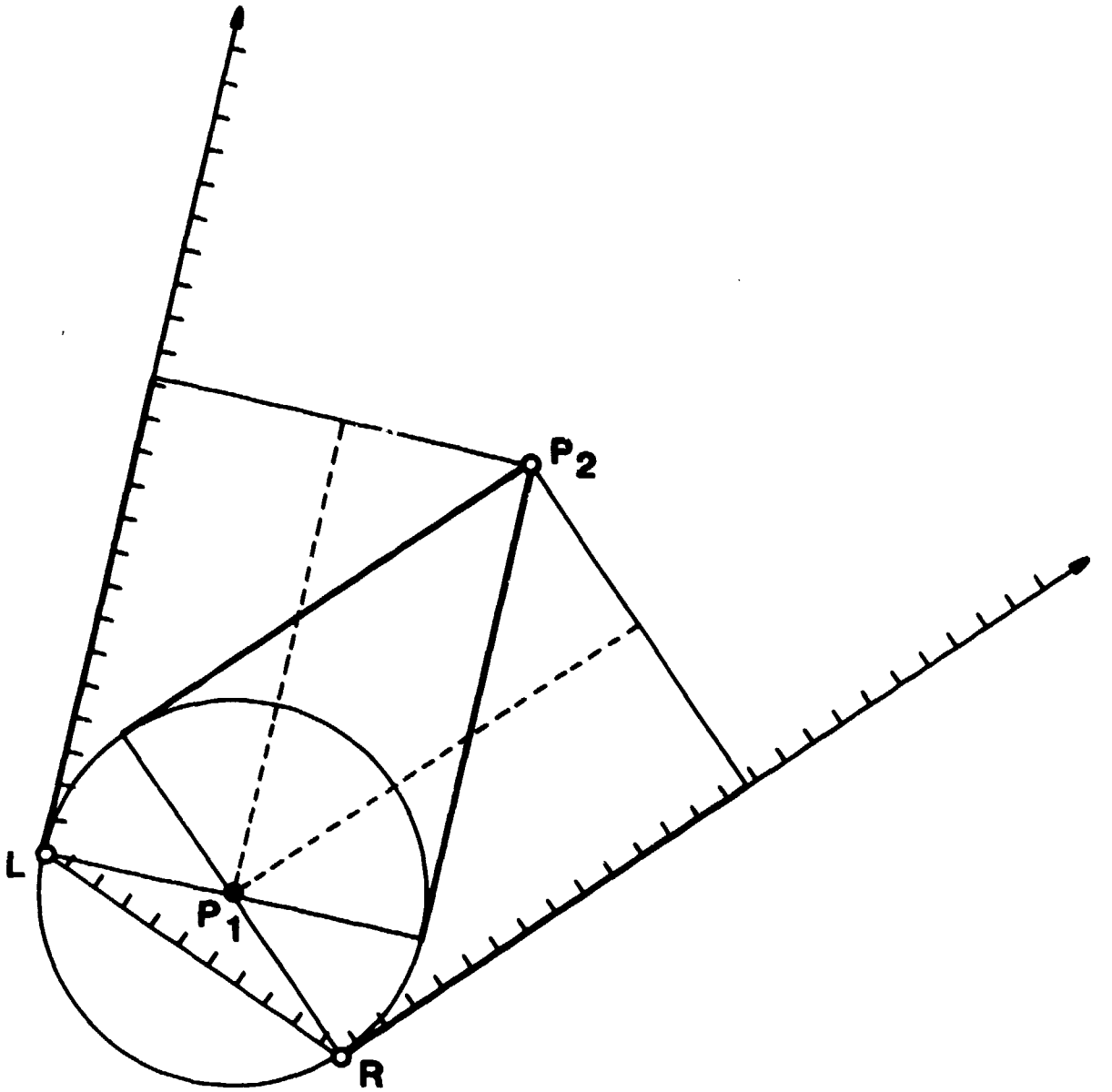

Figure 6. Area of Flexibility Defined by the Two Extreme Tolerance Bands Anchored at P_1 and Containing P_2.

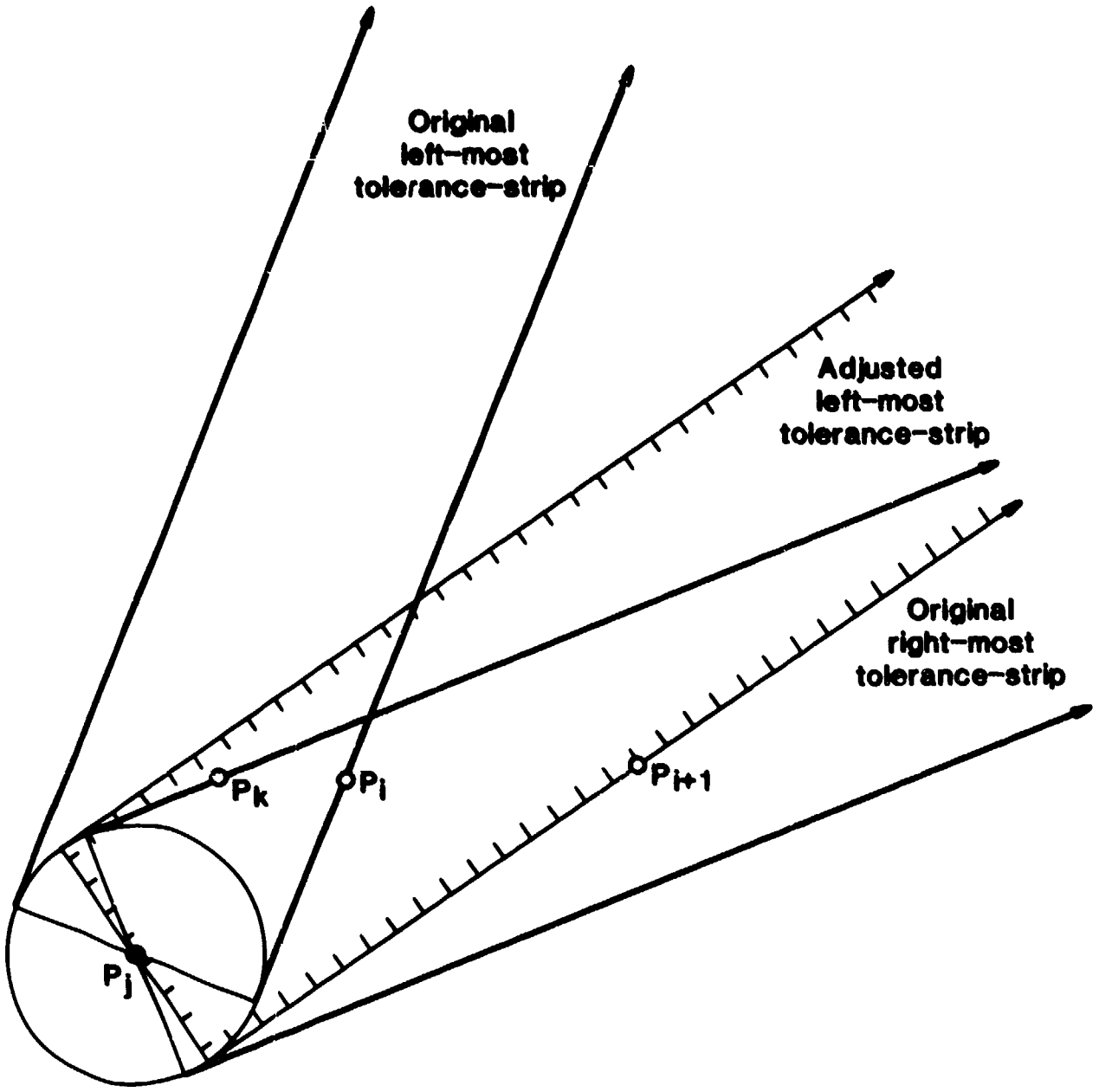

Figure 7. CASE (3): The Subsequent Point P_{i+1} Lies in the Right-Most Tolerance Strip of the Initial Points $\{P_j, P_{j+1}, \ldots, P_i\}$ but not in the Left-Most.

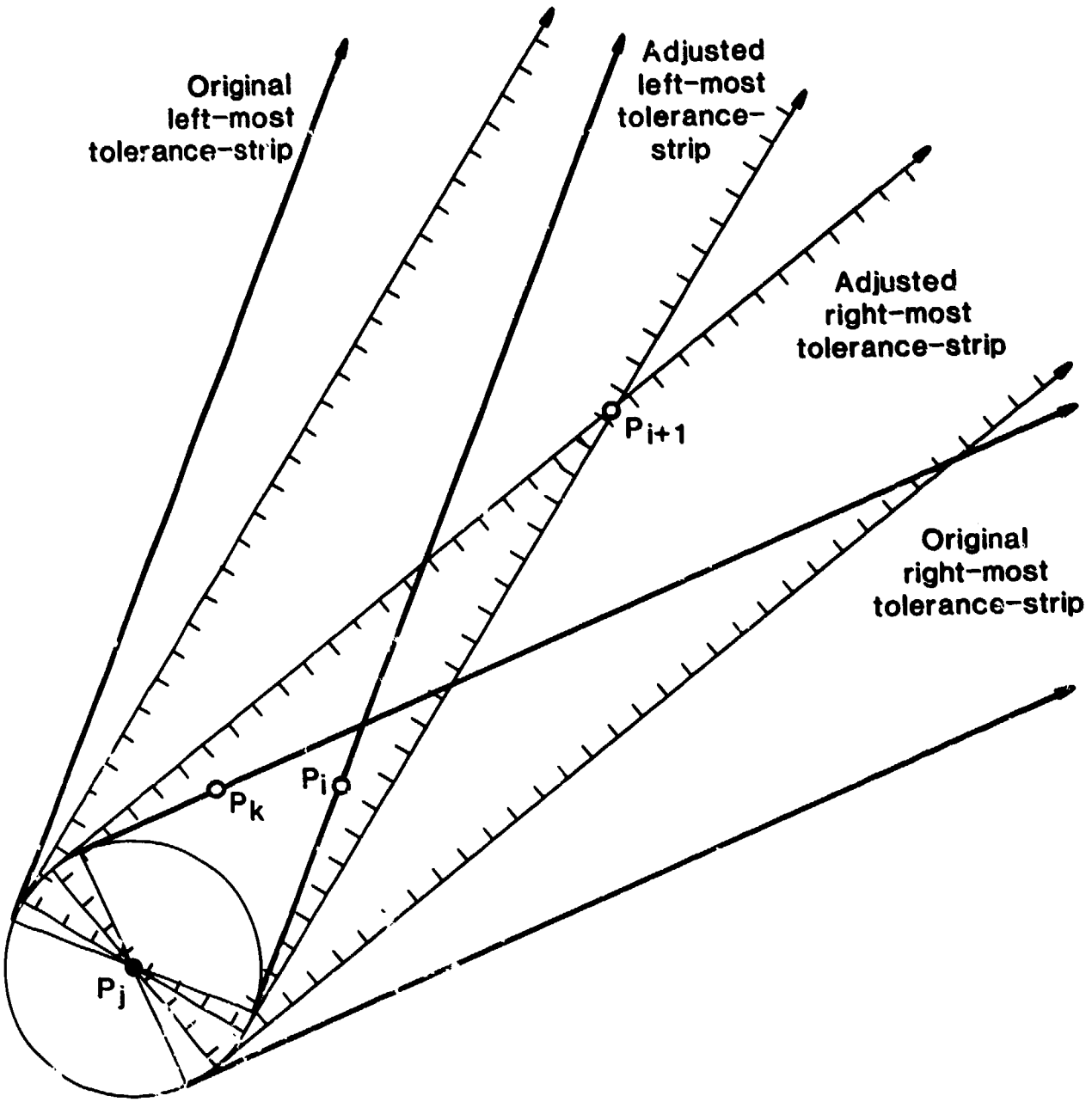

Figure 8. CASE (4): The Subsequent Point P_{i+1} Does Not Lie in any of the Two Extremal Tolerance Strips of the Initial Points $\{P_j, P_{j+1}, \ldots, P_i\}$.

3.3 DETERMINING CONTOUR TANGENTS

Consider three subsequent contour points:

$$(x_1,y_1),\ (x_2,y_2),\ (x_3,y_3).$$

We define the tangent to the contour at the second point as the tangent to the circle through these three points (Figure 9). A vector normal to the tangent at (x_2,y_2) then is given by

$$(3.3.1)\qquad \left(\begin{vmatrix} y_{12} & y_{32} \\ r_{12} & r_{32} \end{vmatrix}, -\begin{vmatrix} x_{12} & x_{32} \\ r_{12} & r_{32} \end{vmatrix} \right)$$

where

$$x_{12} = x_1 - x_2, \qquad x_{32} = x_3 - x_2,$$

$$y_{12} = y_1 - y_2, \qquad y_{32} = y_3 - y_2,$$

$$r_{12} = x_{12} + y_{12}, \qquad r_{32} = x_{32} + y_{32}.$$

This formula is used to calculate an apriori estimate of the normal to the contour tangent at each point that lies in the interior of a contour segment. Note that the terrain gradient is parallel to this normal. The same will hold for the gradient of the subsequently generated synthetic surface. This will ensure a more faithful representation of the terrain.

At the endpoints of contour lines, the tangent was left undetermined. Also if the three points through which the circle is to be passed form an acute angle at the second point, then the above determination of the contour tangent has little meaning. In most of the later runs, the tangent was therefore left undetermined at such points.

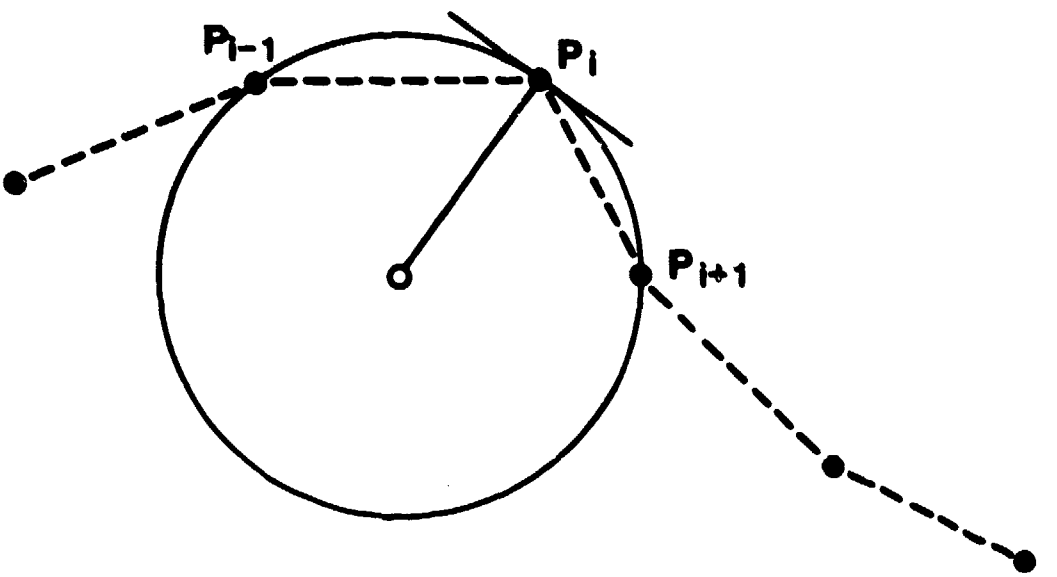

Figure 9. Determining the Tangent at a Contour Point.

4. VORONOI TRIANGULATION OF LARGE SETS OF CONTOUR POINTS

After a suitable sample of the contour points has been selected, the map area is partitioned into triangles whose corners are sample points. We speak of a "triangulation" of the map area. As will be described in the next section, a synthetic surface will be generated by defining a surface patch or "element" in each triangle of the triangulation. For this purpose, a triangulation method that captures the proximity relationships in the sample set is desired. One such method is the Voronoi triangulation, also called Delaunay triangulation, which is obtained from the Voronoi diagram by dualization in a fashion described below.

4.1 DEFINITIONS AND ALGORITHMS

Consider a finite set S of points in the plane. For any point P in S the "Voronoi polygon of P relative to S" is the set of all points in the plane, such that P is as close to any point in this set as is any other point in S. The Voronoi polygon of a point P in S is the intersection of the half-planes which contain P and which are determined by the perpendicular bisectors of the line segments connecting P and the other points in S. Thus, the Voronoi polygon of a point P is a convex polygon, possibly unbounded, which contains P in its interior. Given an arbitrary point X in the plane and the Voronoi polygons associated with a set S, then one and only one of the following statements is true:

(1) X lies in the interior of one and only one Voronoi polygon.
(2) X lies in the interior of an edge shared by two Voronoi polygons.
(3) X is a vertex of three or more Voronoi polygons.

It follows that the Voronoi polygons of the points in S cover the plane without overlapping, that is, without common interior points. The union of their edges forms a diagram, the "Voronoi diagram for S", which partitions the plane into

the Voronoi polygons. A point that lies in the plane and satisfies condition (3) above is said to be a "vertex" of the Voronoi diagram for S. It is called a "degenerate vertex" whenever it is a vertex of more than three Voronoi polygons. Figure 10 illustrates a Voronoi diagram that has a degenerate vertex.

The "dual Voronoi diagram" for a finite set S of points in the plane is the diagram obtained by connecting with straight-line segments those pairs of points in S whose Voronoi polygons relative to S have an edge in common. Figure 11 shows how such a dual diagram is obtained from a Voronoi diagram. The dual diagram defines a collection of non-overlapping convex polygons which cover the convex hull of S. Since each edge of the Voronoi diagram is a line segment whose end-points are vertices of the Voronoi diagram, it follows that there is a one-to-one correspondence between the vertices of the Voronoi diagram and the polygons determined by the dual diagram. In fact, it follows from the definition of a Voronoi polygon that a vertex of the Voronoi diagram is equidistant from the points in S that are vertices of the corresponding polygon determined by the dual diagram, and it is closer to these points than it is to any other point in S. In general, most vertices of the Voronoi diagram are non-degenerate, so that most of the polygons defined by the dual diagram are triangles. In the presence of degenerate vertices, the corresponding polygons determined by the dual diagram can be partitioned into non-overlapping triangles by introducing suitable diagonals. Thus a "Voronoi triangulation" results from the dual diagram for the set S. We note that, while the dual Voronoi diagram is uniquely determined, there are several compatible Voronoi triangulations in the presence of degeneracies. Figures 12 and 13, respectively, illustrate the dual diagram and a Voronoi triangulation that results from it.

J. Bernal and S. E. Howe of the National Bureau of Standards (NBS) have generalized, extended and combined algorithms by Bentley, Weide and Yao (80), and Bowyer (81), to obtain an algorithm which constructs the Voronoi diagram and, therefore, a Voronoi triangulation, in "linear expected time" for a set S of points distributed uniformly in the interior of a rectangle in the plane. This material is being readied for publication under the title "Expected O(N) and $O(N^{4/3})$ Algorithms for Constructing Voronoi Diagrams in Two and Three

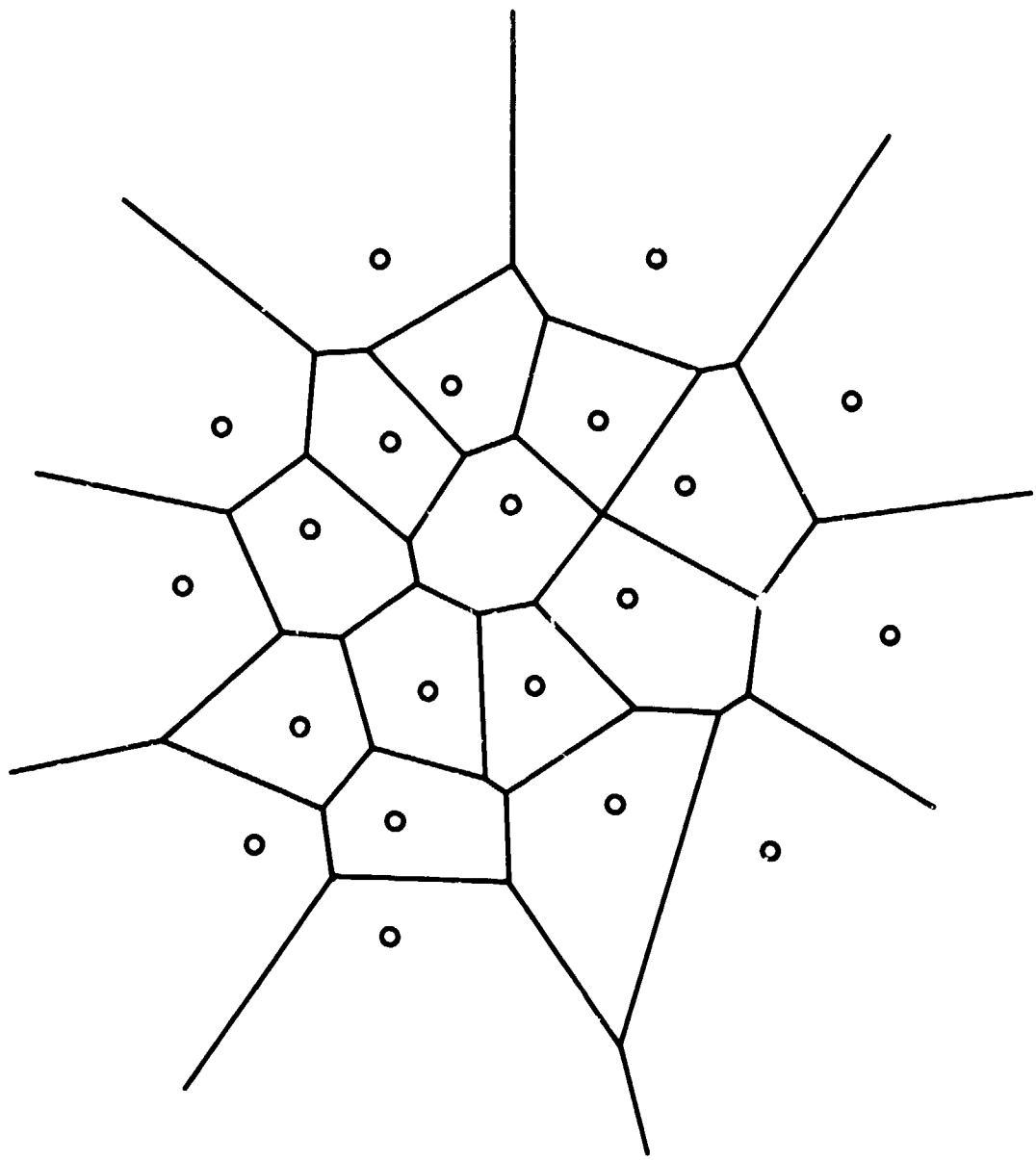

Figure 10. Voronoi Diagram of Point Set Indicated by Small Circles.

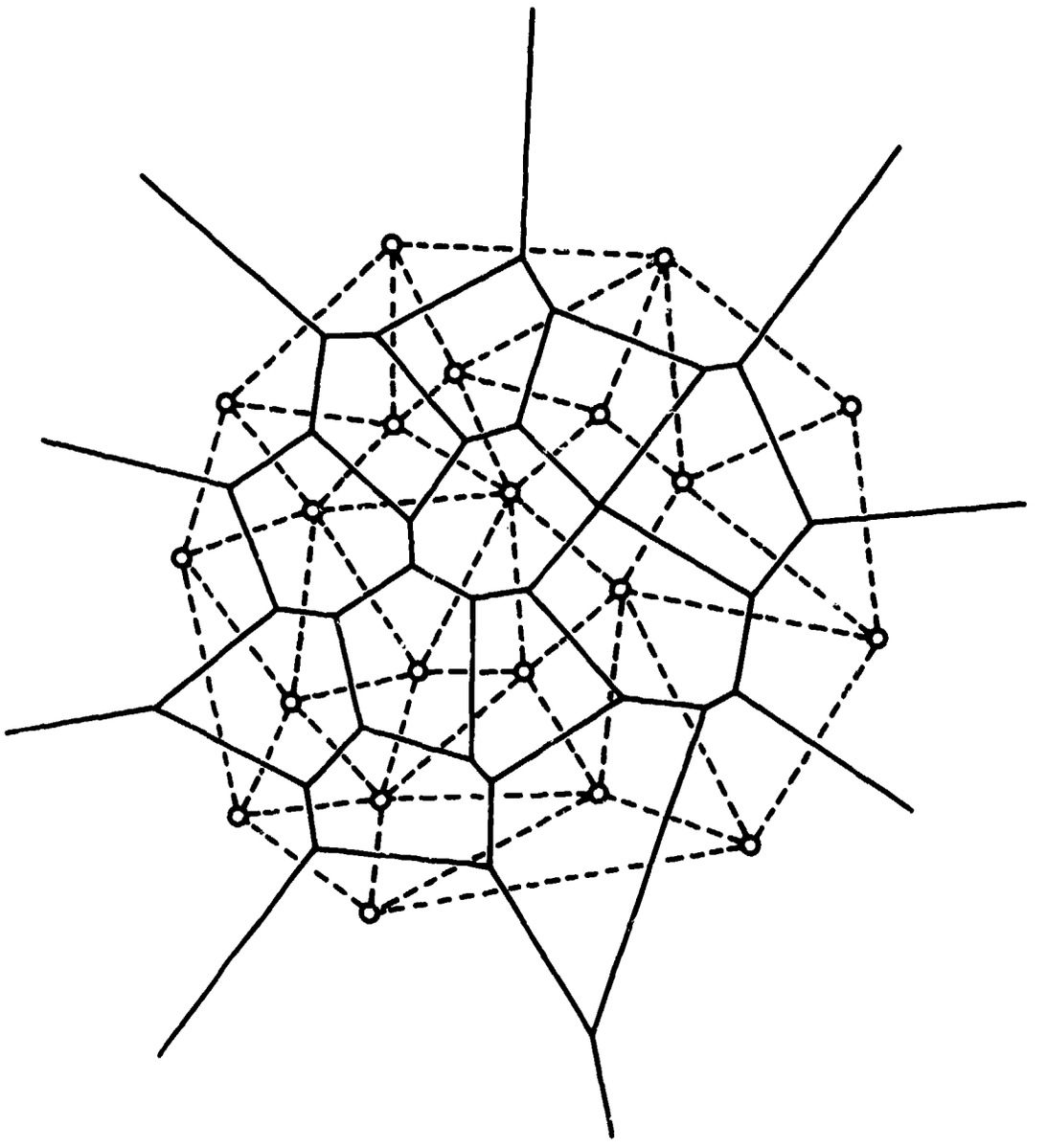

Figure 11. Voronoi Diagram Supplemented by its Dual.

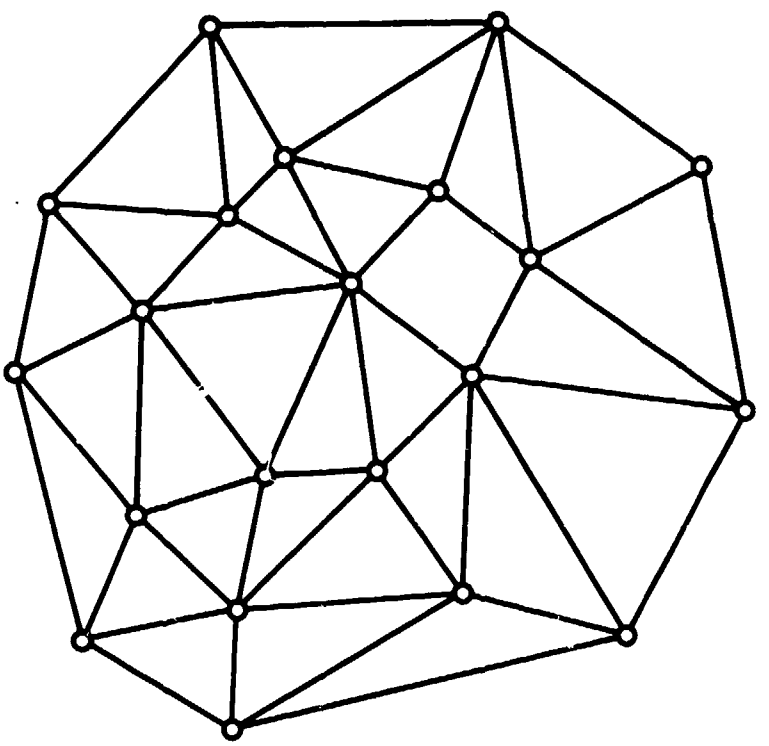

Figure 12. The Dual Voronoi Diagram or Delaunay Diagram Representing Neighbor Relations Between Points.

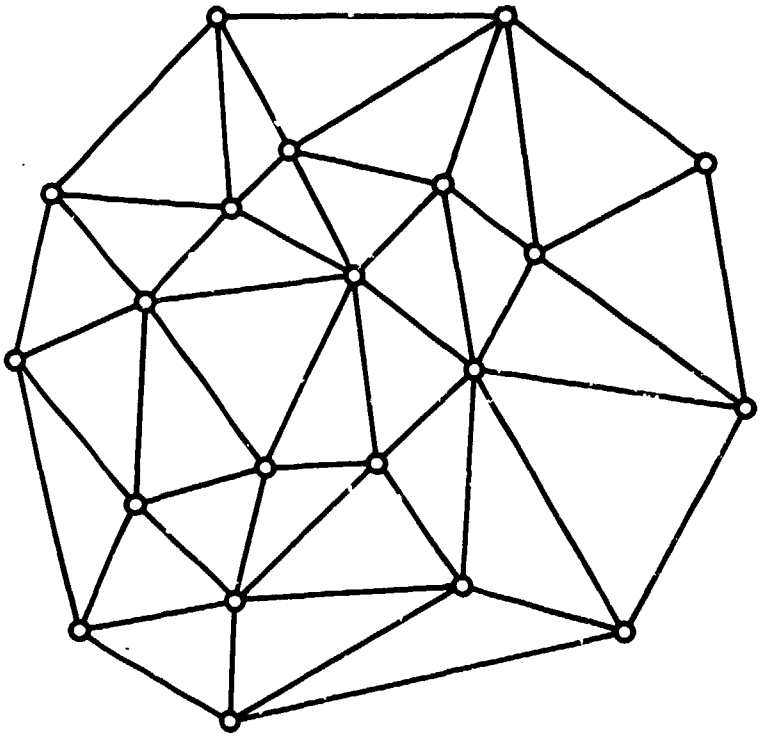

Figure 13. Voronoi Triangulation Obtained by Adding a Diagonal to a Non-Triangular Cell in the Dual Diagram in Figure 12.

Dimensions" by J. Bernal and S. E. Howe. A draft copy is available upon request.

In the resulting algorithm, the set S which is to be triangulated is enclosed into a rectangle. The algorithm consists of three steps. The first step divides the rectangle into approximately N (number of points in S) equally sized square cells, and assigns each point in S its proper cell. On the average there will be one point per cell, although there may be empty cells as well as cells with more than one point. Each cell is of the form

$$\{(x, y): x \leq x \leq x+b, \quad y \leq y \leq y+b\},$$

for some x, y, b, and a point in S is assigned that cell if it lies in that area. Cells within a distance of two cells from the boundary of the rectangle are called "outer cells" and all others, "inner cells."

The second step constructs the Voronoi polygons of the points belonging to inner cells. Given a point P in an inner cell, a search for other points in S is conducted through each of the layers of cells surrounding P. This search procedure, called a "spiral search", starts with the cell that contains P, and then proceeds in outward direction to each of the layers of cells surrounding this cell. The Voronoi polygon of P is progressively built by intersecting the half planes which contain P and which are determined by the perpendicular bisectors of the line segments connecting P and the points in S found through the search. A geometrical test is available, which permits to ascertain whether the Voronoi polygon has achieved its final form. In most cases the Voronoi polygon of P is obtained after examining only a small number of cells and points.

The third step, finally, builds the Voronoi polygons of points in the outer cells by applying a modified version of Bowyer's insertion algorithm to this set of points.

4.2 IMPLEMENTATION

J. Bernal and S. E. Howe have implemented the above algorithm on a Control Data Cyber 205 at NBS. The implementation consists of about 5,000 FORTRAN statements. It requires as input a tolerance ε, and a list of the x and y coordinates of the points in the set for which a Voronoi diagram is desired. This list of points must be free of duplication, that is, the distance between any two points must always be above the tolerance ε. The execution of the package requires approximately 34N words of memory, where N is the number of points to be triangulated. Versions of this package were successfully transferred first to a VAX 11/750 and then to a VAX 11/780 at ETL. This required adaptations which are described below.

One of the main objectives of our work was to demonstrate the feasibility of triangulating data sets of about 40,000 to 70,000 points. Two difficulties arose in the course of the demonstration. The first difficulty was that of furnishing the package with the ability to deal with data sets whose members are not all necessarily distinct. The second difficulty had to do with memory restrictions that would not allow the execution of the package for more than 50,000 points at a time.

The "check for duplication" posed a difficulty because it was not possible to consider every pair of points in view of their large number. In the case of a 40,000 point set this would have amounted to the examination of about 800,000,000 pairs. Fortunately, it was discovered that a portion of the Voronoi triangulation package already provided a useful tool for the solution of this problem, namely the cell structure set up by the implementation of the first step of the algorithm. Thus, a procedure was developed which takes advantage of this structure to check for the duplication of points. Given a point P in a set S, use a spiral search through each of the layers of cells surrounding P to search for other points in S. Eliminate any point found through the search whose distance from P does not exceed the tolerance ε.

Terminate the search as soon as every cell that intersects the closed circular disk with center P and radius has been searched. In general, the disk defined above is contained in the cell that contains P, so that this is the only cell that has to be searched. This procedure was incorporated into the package, and was able to identify and remove duplications for data sets of 25,000 points in less than 30 seconds of CPU time.

The restrictions on available memory mentioned above required the development of a "decomposition procedure" which allows the separate triangulation of a finite number of subsets of a data set in such a way that the correct total triangulation results. This procedure will now be described.

Given a positive integer k we select numbers x_0, x_k, y_0, y_1, such that $x_0 < x_k$, $y_0 < y_1$, and the rectangle

$$R = \{(x, y): x_0 \leq x \leq x_k, \quad y_0 \leq y \leq y_1\},$$

contains the data set. We select numbers

$$x_{0L}, x_{1L}, x_1, x_{1R}, x_{2L}, x_2, x_{2R}, \cdots, x_{k-1,L}, x_{k-1}, x_{k-1,R}, x_{kR},$$

such that

$$x_0 < x_1 < \ldots < x_{k-1} < x_k \text{ and}$$

$$x_0 = x_{0L}, x_{1L} < x_1 < x_{1R}, x_{2L} < x_2 < x_{2R}, \cdots, x_{k-1,L} < x_{k-1} < x_{k-1,R}, x_{kR} = x_k.$$

For each i, i=1, 2,..., k, we define rectangles R_i' and R_i by:

$$R_i' = \{(x, y): x_{i-1,L} \leq x \leq x_{i,R}, \quad y_0 \leq y \leq y_1\},$$

$$R_i = \{(x, y): x_{i-1} \leq x \leq x_i, \quad y_0 \leq y \leq y_1\}.$$

It follows that $R_i \subset R_i'$ for each i, i=1, 2,..., k, and $R = \bigcup_{i=1}^{k} R_i = \bigcup_{i=1}^{k} R_i'$.

We also assume that the points (x_0, y_0), (x_0, y_1), (x_1, y_0), (x_1, y_1),..., (x_{k-1}, y_0), (x_{k-1}, y_1), (x_k, y_0), (x_k, y_1), that is, the corner points of the rectangles R_i, i=i,..., k, belong to the data set (Figure 14). Furthermore, we define S_i to be the set of points in the data set that belong to R_i' for each i, i=1,..., k. Assuming that k and the numbers x_{0L}, x_{1L}, x_1, x_{1R},..., $x_{k-1,L}$, x_{k-1}, $x_{k-1,R}$, x_{kR} have been properly selected, the decomposition procedure consists of obtaining separate triangulations T_i, i=1,..., k, for the sets S_i, i=1,..., k, respectively (Figure 15). The correct triangulation of the entire data set is then given by:

$$\bigcup_{i=1}^{k} \{ t \in T_i : R_i \text{ intersects the interior of } t \}.$$

In order to properly select k and the numbers x_{0L}, x_{1L}, x_1, x_{1R},..., $x_{k-1,L}$, x_{k-1}, $x_{k-1,R}$, x_{kR}, a separate procedure was developed. In what follows, we define, for a given triangle t in a Voronoi triangulation, x(t) and y(t) to be the x,y-coordinates of the vertex in the Voronoi diagram that corresponds to t. Accordingly, we define d(t) to be the distance from (x(t), y(t)) to any one of the vertices of t. Since (x(t), y(t)) is equidistant from the vertices of t, d(t) is well defined. In the following procedure, m denotes the maximum number of points that can be triangulated with a single run of the package:

Step 1. Let k = 1 and obtain R_1, R_1' and S_1. Let j = 1.

Step 2. If the number of points in S_j does not exceed m go to step 3. Else increase k to the next positive integer for which x_{0L}, x_{1L}, x_1, x_{1R},..., $x_{k-1,L}$, x_{k-1}, $x_{k-1,R}$, x_{kR}, can be defined with: $x_0<x_1<\ldots<x_{k-1}<x_k$; $x_{1L}<x_1<x_{1R}$, ..., $x_{k-1,L}<x_{k-1}<x_{k-1,R}$; $x_0=x_{0L}$; $x_{kR}=x_k$; and the corresponding R_i, R_i', S_i ..., k, can be obtained with the number of points in each S_i, i=1,..., not exceeding m. Let j = 1.

Step 3. Obtain the Voronoi triangulation T_j for S_j. If j is equal to 1 let $x_L=x_{0L}$. Else define x_L by:

$$x_L = \min \{ x(t) - d(t): t \in T_j, \text{ 2 vertices of } t \text{ lie in } R_{j-1}, \text{ 1 vertex of } t \text{ lies in the interior of } R_j \}.$$

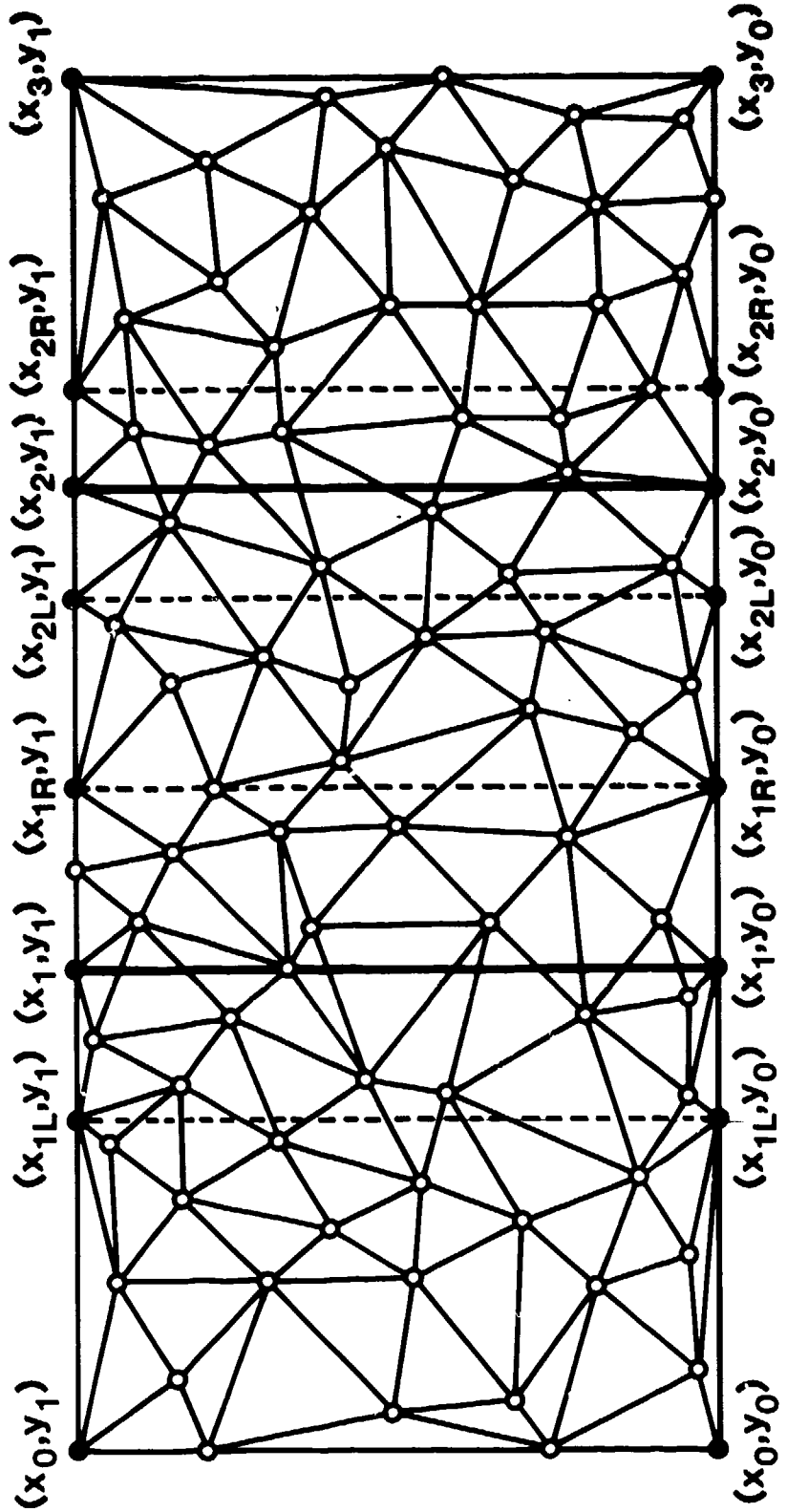

Figure 14. Decomposition of a Rectangular Point Set for Voronoi Triangulation.

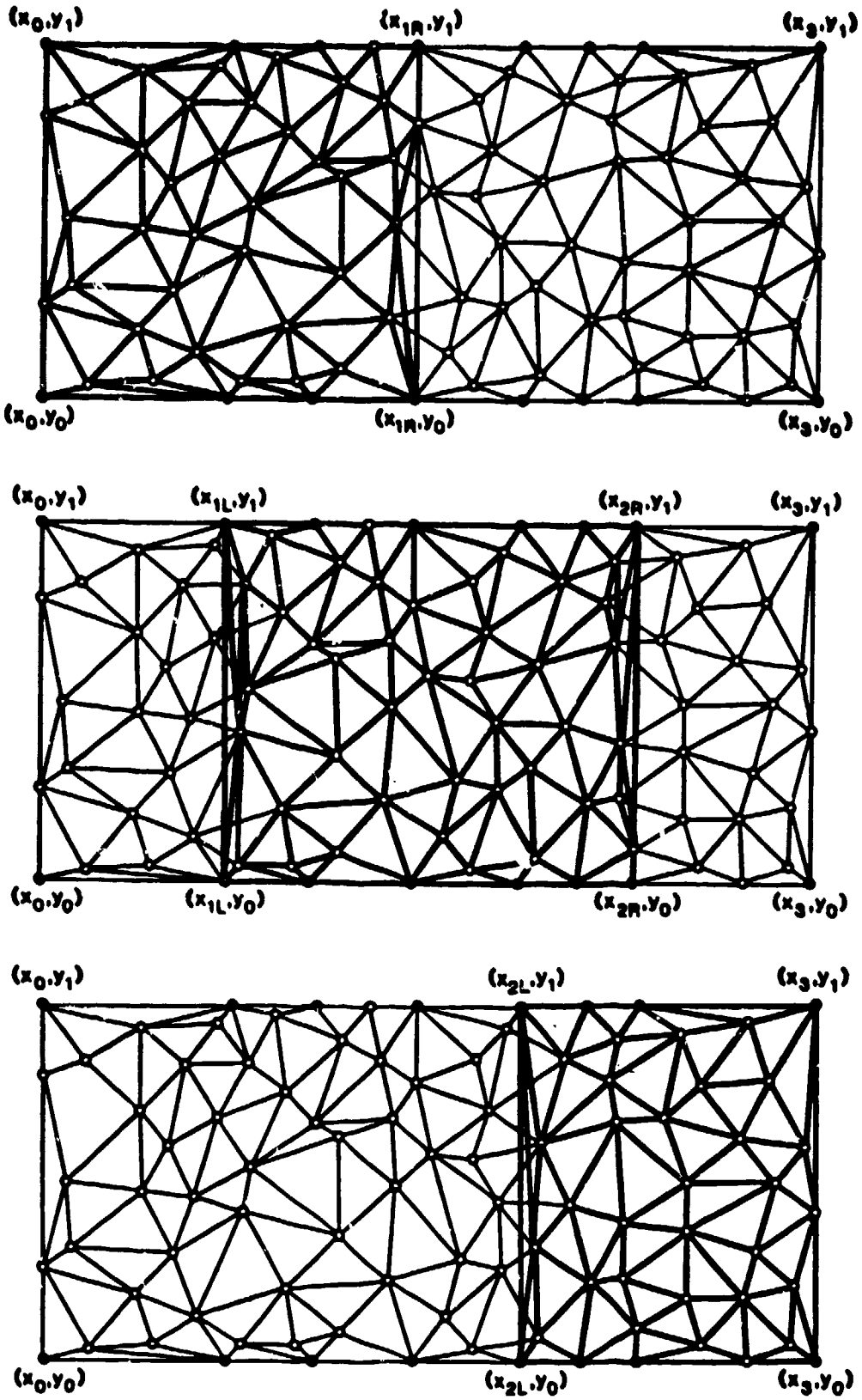

Figure 15. Voronoi Triangulations of Three Sections Superimposed Over the Voronoi Triangulation of the Entire Set.

If j is equal to k, let $x_R = x_{kR}$. Else define x_R by:

$$x_R = \max \{ x(t) + d(t): t \in T_j, \text{ 2 vertices of t lie in } R_{j+1}, \text{ 1 vertex of t lies in the interior of } R_j \}.$$

If $x_{j-1,L}$ does not exceed x_L and x_R does not exceed x_{jR} go to step 4. Else,

if $x_{j-1,L}$ exceeds x_L let $x_{j-1,L} = x_L$, and if x_R exceeds x_{jR} let $x_{jR} = x_R$. Obtain R_j, R_j', S_j and go to step 2.

Step 4. If j equals k, stop. Else let j = j + 1 and go to step 3.

This procedure was incorporated into the triangulation package and is currently operational.

4.3 RESULTS OF A COMPUTATIONAL EXPERIMENT

A Voronoi triangulation package that includes the adaptations described above was implemented at ETL and NBS. The feasibility of performing a triangulation with large sets of data was demonstrated. For a specific experiment conducted on the VAX 11/750, we divided the triangles into three classes:

Class 1: Those whose vertices lie exactly on one contour line of the map.
Class 2: Those whose vertices lie exactly on two contour lines of the map.
Class 3: And those whose vertices lie exactly on three contour lines.

Each class may contain triangles that are intersected by contour lines other than those containing their vertices, and any given triangle belongs to one and only one class.

A first data set contained 39,645 points and was decomposed into two subsets. A second data set contained 70,249 points and was decomposed into four subsets. The tables below illustrate some of the results obtained when triangulating these data sets with the Voronoi triangulation package. The times, given in CPU seconds, indicate the rate at which the package ran per point.

TABLE FOR N = 39,645

Element of Decomposition	% Class 1 Triangles	% Class 2 Triangles	% Class 3 Triangles	Number of Triangles	CPU Time Sec/Point
1	17.20	49.96	32.84	41,875	0.044175
2	17.05	48.09	34.86	36,451	0.046810

TABLE FOR N = 70,249

Element of Decomposition	% Class 1 Triangles	% Class 2 Triangles	% Class 3 Triangles	Number of Triangles	CPU Time Sec/Point
1	22.58	59.58	17.84	41,134	0.048542
2	22.65	58.65	18.70	32,824	0.043350
3	18.74	59.53	21.73	33,715	0.044793
4	26.00	55.76	18.24	31,755	0.052509

5. SURFACE GENERATION WITH CLOUGH-TOCHER ELEMENTS

In this section we discuss the construction of a surface function passing through irregularly spaced given points for most of which both elevations and contour tangents are specified. The planar projections of all given points are triangulated, that is, the map area is tiled with triangles whose vertices are these points. The description of the surface is in terms of these triangles: in order to find the surface elevation for an arbitrary point in the map region, a triangle containing this point must be found. An evaluation formula or "element" is then invoked using triangle-specific parameters. The "Clough-Tocher element" employed in our work requires the determination of suitable tangent planes at the given points. For this purpose, "local" as well as more expensive "global" methods are available. A global method based on energy minimization has been implemented and tested for computational feasibility.

5.1 THE CLOUGH-TOCHER ELEMENT

Triangulation-based surface interpolation is a classical computational problem. Various versions of the Finite Element Method (see Zienkiewicz (71), Birkhoff and Mansfield (74)) are usually employed in its solution. The "linear element" represents linear interpolation by the plane through the vertices of the triangle at hand. It yields a surface of continuous elevation. However, this surface is not smooth since "creases", that is, tangential discontinuities, occur along the boundaries of the triangles.

Nonlinear elements are needed for smooth surface interpolation. A major advantage of smooth surfaces is that their corresponding surface functions are uniquely differentiable at each point of their domain; in other words, there are unique gradients. The greater flexibility and information content of nonlinear elements also allows for a more precise representation of the original data than that provided by linear elements, given triangulations of comparable densities.

The "Clough-Tocher" element (see Clough and Tocher (65), Lawson (72, 76, 77)) is a particularly attractive tool for smooth surface interpolation. It requires that at each vertex i of the triangle over which it is defined, the elevation z_i and the partial derivatives z_{ix} and z_{iy} be given. The Clough-Tocher element then is described by a function $z = f(x,y)$ on the given planar triangle. It represents a "surface patch" above this triangle (Figure 17, see also Figure 16). The surface patch meets the prescribed elevation as well as the prescribed derivatives at each vertex. It is fully defined by these quantities, in other words, by the three elevations and the three tangential planes (gradients).

The following considerations concern the construction of an entire surface from such surface patches. In order to ensure continuity of elevation between adjacent triangles, the Clough-Tocher element satisfies the following

(5.1.1) Cubic Boundary Condition: Along each triangle edge, the Clough-Tocher element agrees with a cubic (degree 3 or less) polynomial in terms of a variable linearly traversing the edge.

A cubic polynomial in one variable is completely determined by two elevations and two derivatives (in the direction of the edge). Since the tangential planes at the vertices agree, so do the derivatives in the direction of the edge. Therefore, the Clough-Tocher elements of two adjacent triangles determine the same cubic polynomial on the edge they share. It follows that not only the elevations, but also the derivatives in the direction of the edge agree along that common boundary.

In order to ensure smoothness across the triangle boundaries, the Clough-Tocher element satisfies the following

(5.1.2) Linear Derivative Condition: Along each triangle edge, the derivative taken in the direction perpendicular to the edge varies linearly between the values it assumes at the ends of the edge.

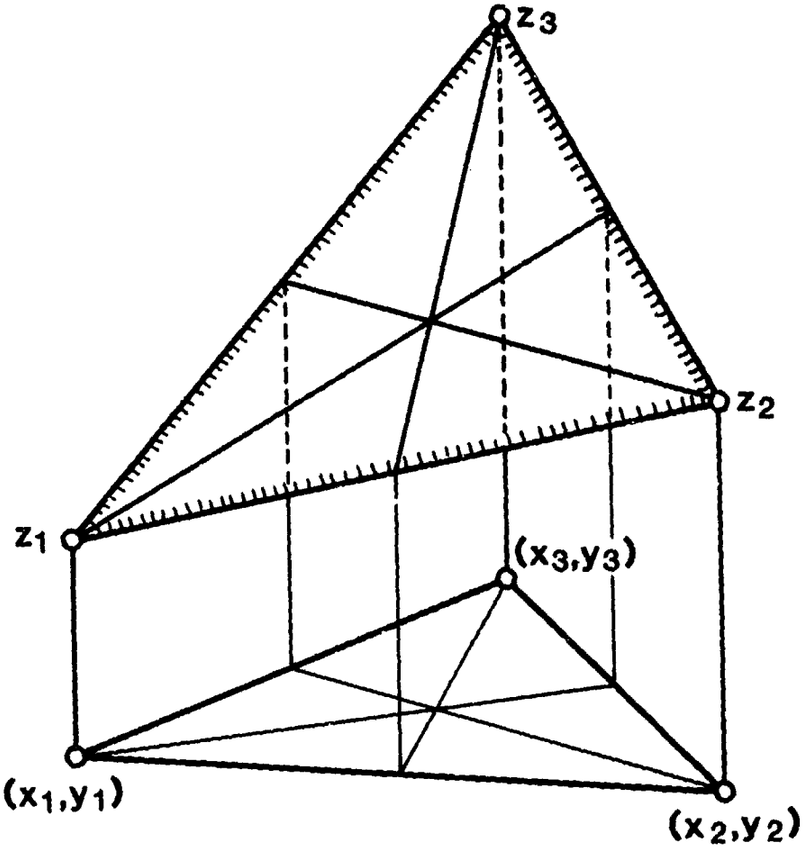

Figure 16. Linear Element.

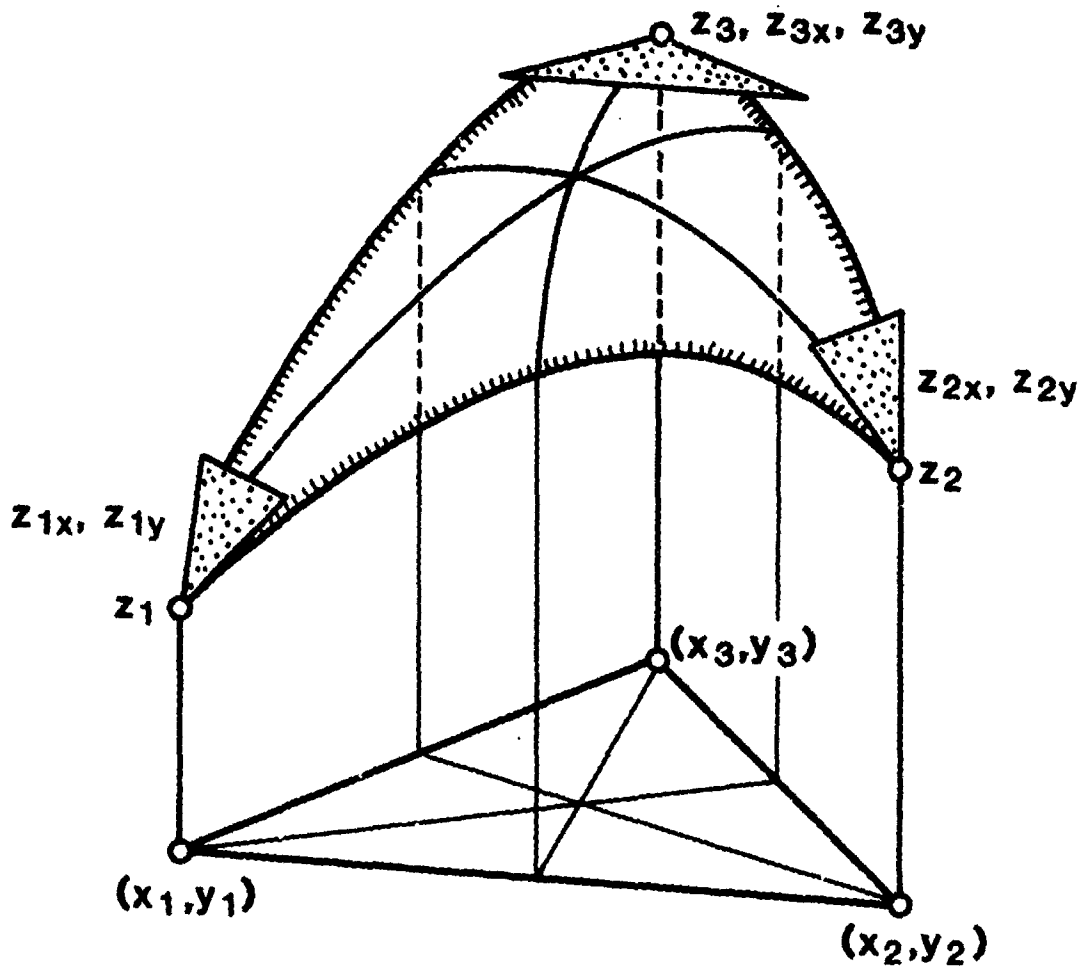

Figure 17. Clough-Tocher Element.

The edge-perpendicular derivatives along an edge are thus uniquely determined by their location on the edge and the derivatives at the vertices at the ends of the edge. Since the latter agree for two adjacent triangles, the edge-perpendicular derivatives along their common edge agree also. It was seen that, as a consequence of the cubic boundary condition, the edge-parallel derivatives agree. Hence, adjacent Clough-Tocher elements share tangential planes everywhere along their common boundary.

Functions over triangles are best expressed in terms of their "barycentric coordinates", also called "triangle coordinates." These are three real numbers

$$\lambda_1, \qquad \lambda_2, \qquad \lambda_3,$$

such that

$$(5.1.3) \quad \lambda_1 + \lambda_2 + \lambda_3 = 1$$

$$\lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 = x$$

$$\lambda_1 y_1 + \lambda_2 y_2 + \lambda_3 y_3 = y,$$

where x, y are the planar coordinates of the point in question. The barycentric coordinates are functions of these planar coordinates. To express these functional relationships, we use Zienkiewicz notation:

$$x_{ij} = x_i - x_j, \qquad y_{ij} = y_i - y_j, \qquad i,j = 1, 2, 3.$$

We then have

$$(5.1.4) \quad \lambda_1 = \lambda_1(x,y) = (y_{23}\cdot x + x_{32}\cdot y + x_2 y_3 - y_2 x_3)/D$$

$$\lambda_2 = \lambda_2(x,y) = (y_{31}\cdot x + x_{13}\cdot y + x_3 y_1 - y_3 x_1)/D$$

$$\lambda_3 = \lambda_3(x,y) = (y_{12}\cdot x + x_{21}\cdot y + x_1 y_2 - y_1 x_2)/D,$$

where the denominator is given by the determinant

$$D = \begin{vmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{vmatrix}.$$

We also note that the partial derivatives of the barycentric coordinates with respect to x, y, are given by

$$(5.1.5) \quad \lambda_{1x} = y_{23}/D, \quad \lambda_{2x} = y_{31}/D, \quad \lambda_{3x} = y_{12}/D,$$

$$\lambda_{1y} = x_{32}/D, \quad \lambda_{2y} = x_{13}/D, \quad \lambda_{3y} = x_{21}/D.$$

The advantage of barycentric coordinates lies in the symmetric way the vertices of the triangle are treated. Also, their signs indicate immediately whether the point (x,y) lies inside or outside the triangle: a negative barycentric coordinate indicates that the point lies outside the triangle. If all barycentric coordinates are positive, then the point lies in the interior of the triangle. On the boundary at least one barycentric coordinate vanishes. Vertices are characterized by single nonzero barycentric coordinates of value 1:

$$(1, 0, 0) , (0, 1, 0) , (0, 0, 1).$$

The barycenter of the triangle or "centroid" is given by

$$(1/3, 1/3, 1/3).$$

It defines what we call a "barycentric partition" (Figure 18) of the triangle

$$(5.1.6) \quad T = B_1 \cup B_2 \cup B_3 \cup B_0,$$

where

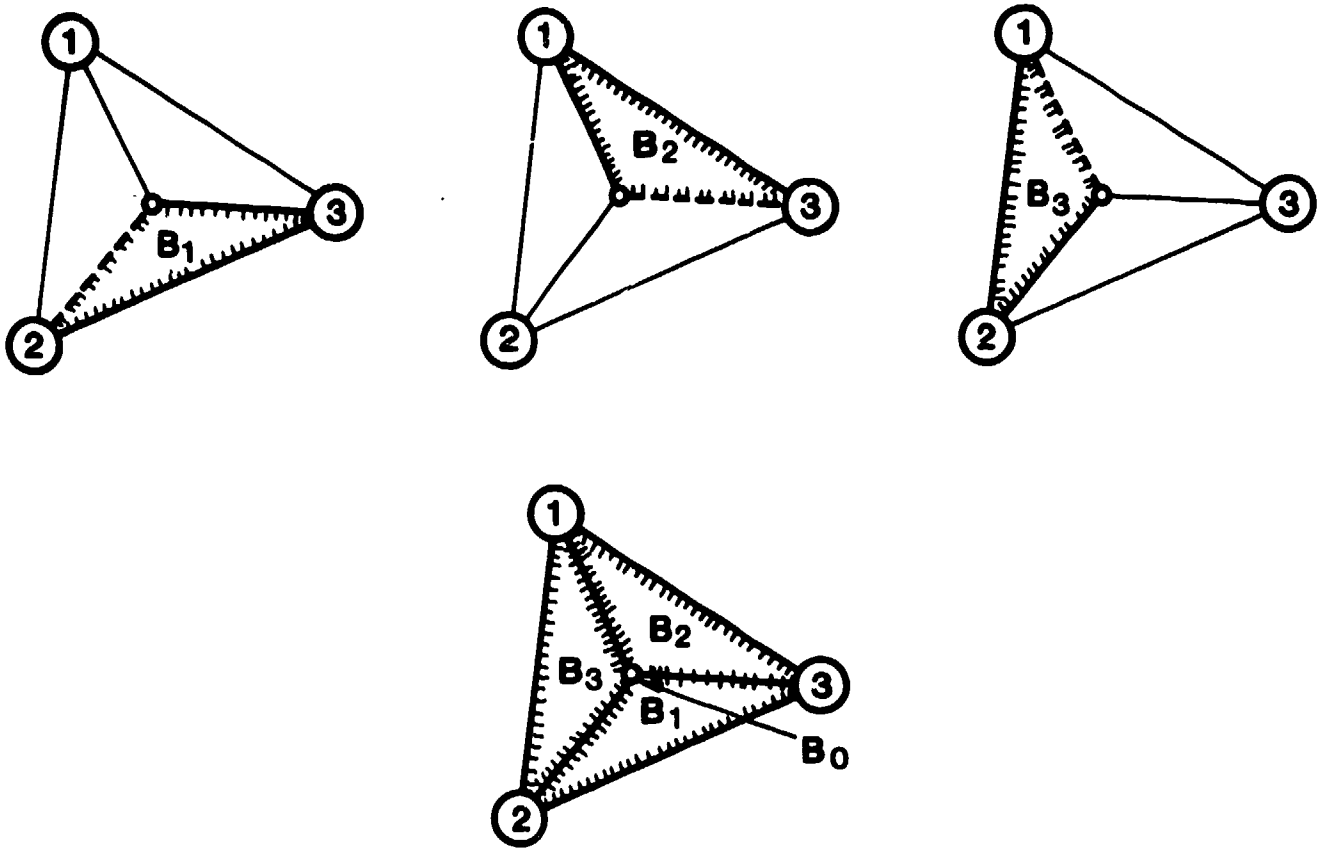

Figure 18. Barycentric Partition of Triangle.

$$B_1 = \{(\lambda_1,\lambda_2,\lambda_3):\ \lambda_1 < \lambda_2,\ \lambda_1 \le \lambda_3\}$$

$$B_2 = \{(\lambda_1,\lambda_2,\lambda_3):\ \lambda_2 < \lambda_3,\ \lambda_2 \le \lambda_1\}$$

$$B_3 = \{(\lambda_1,\lambda_2,\lambda_3):\ \lambda_3 < \lambda_1,\ \lambda_3 \le \lambda_2\}$$

$$B_0 = \{(\lambda_1,\lambda_2,\lambda_3):\ \lambda_1 = \lambda_2 = \lambda_3 = 1/3\}.$$

In each of the major triangle regions B_i, $i = 1,2,3$, the corresponding barycentric coordinate is dominated by the remaining ones:

$$\lambda_i = \min\{\lambda_1, \lambda_2, \lambda_3\}.$$

Each function representing a Clough-Tocher element is a cubic polynomial of the barycentric coordinates in each of the major regions B_i of the barycentric partition of the triangle. The function is continous and smooth at the boundaries of these regions of the barycenter. We call such a function "piecewise cubic" with respect to the barycentric partition.

In his seminal work, Lawson (76) introduces three "correction functions"

$$(5.1.8)\quad \varphi_i = \varphi_i(\lambda_1,\lambda_2,\lambda_3),\quad i = 1,2,3,$$

with which to describe the Clough-Tocher element. They are piecewise cubic with respect to the barycentric partition, and given by

$$\varphi_1 = \begin{cases} \lambda_1\lambda_2\lambda_3 + 5/6\lambda_1^3 - 1/2\lambda_1^2 & \text{for } (\lambda_1,\lambda_2,\lambda_3) \in B_1 \\ -1/6\lambda_2^3 + 1/2\lambda_2^2\lambda_3 & \text{for } (\lambda_1,\lambda_2,\lambda_3) \in B_2 \\ -1/6\lambda_3^3 + 1/2\lambda_3^2\lambda_2 & \text{for } (\lambda_1,\lambda_2,\lambda_3) \in B_3 \\ 1/81 & \text{for } (\lambda_1,\lambda_2,\lambda_3) \in B_0 \end{cases}$$

$$\rho_2 = \begin{cases} \lambda_1\lambda_2\lambda_3 + 5/6\lambda_2^3 - 1/2\lambda_2^2 & \text{for } (\lambda_1,\lambda_2,\lambda_3) \in B_2 \\ -1/6\lambda_3^3 + 1/2\lambda_3^2\lambda_1 & \text{for } (\lambda_1\ \lambda_2,\lambda_3) \in B_3 \\ -1/6\lambda_1^3 + 1/2\lambda_1^2\lambda_3 & \text{for } (\lambda_1,\lambda_2,\lambda_3) \in B_1 \\ 1/81 & \text{for } (\lambda_1,\lambda_2,\lambda_3) \in B_0 \end{cases}$$

$$\rho_3 = \begin{cases} \lambda_1\lambda_2\lambda_3 + 5/6\lambda_3^3 - 1/2\lambda_3^2 & \text{for } (\lambda_1,\lambda_2,\lambda_3) \in B_3 \\ -1/6\lambda_1^3 + 1/2\lambda_1^2\lambda_2 & \text{for } (\lambda_1,\lambda_2,\lambda_3) \in B_1 \\ -1/6\lambda_2^3 + 1/2\lambda_2^2\lambda_1 & \text{for } (\lambda_1,\lambda_2,\lambda_3) \in B_2 \\ 1/81 & \text{for } (\lambda_1,\lambda_2,\lambda_3) \in B_0. \end{cases}$$

The motivation for the choice of the correction functions is given in Lawson (76). The partial derivatives of these correction functions are:

(5.1.9)

$$D\cdot\rho_{1x} = \begin{cases} (\lambda_2\lambda_3+5/2\lambda_1^2-\lambda_1)y_{23} + \lambda_3\lambda_1 y_{31} + \lambda_1\lambda_2 y_{12} & \text{in } B_1 \\ (-1/2\lambda_2^2+\lambda_2\lambda_3)y_{31} + 1/2\lambda_2^2 y_{12} & \text{in } B_2 \\ (-1/2\lambda_3^2+\lambda_3\lambda_2)y_{12} + 1/2\lambda_3^2 y_{31} & \text{in } B_3 \\ -1/18y_{23} & \text{in } B_0 \end{cases}$$

$$D\cdot\rho_{1y} = \begin{cases} (\lambda_2\lambda_3+5/2\lambda_1^2-\lambda_1)x_{32} + \lambda_3\lambda_1 x_{13} + \lambda_1\lambda_2 x_{21} & \text{in } B_1 \\ (-1/2\lambda_2^2+\lambda_2\lambda_3)x_{13} + 1/2\lambda_2^2 x_{21} & \text{in } B_2 \\ (-1/2\lambda_3^2+\lambda_3\lambda_2)x_{21} + 1/2\lambda_3^2 x_{13} & \text{in } B_3 \\ -1/18x_{32} & \text{in } B_0 \end{cases}$$

$$D\cdot\rho_{2x} = \begin{cases} (\lambda_3\lambda_1+5/2\lambda_2{}^2-\lambda_2)y_{31} + \lambda_1\lambda_2 y_{12} + \lambda_2\lambda_3 y_{23} & \text{in } B_2 \\ (-1/2\lambda_3{}^2+\lambda_3\lambda_1)y_{12} + 1/2\lambda_3{}^2 y_{23} & \text{in } B_3 \\ (-1/2\lambda_1{}^2+\lambda_1\lambda_3)y_{23} + 1/2\lambda_1{}^2 y_{12} & \text{in } B_1 \\ -1/18 y_{31} & \text{in } B_0 \end{cases}$$

$$D\cdot\rho_{2y} = \begin{cases} (\lambda_3\lambda_1+5/2\lambda_2{}^2-\lambda_2)x_{13} + \lambda_1\lambda_2 x_{21} + \lambda_2\lambda_3 x_{32} & \text{in } B_2 \\ (-1/2\lambda_3{}^2+\lambda_3\lambda_1)x_{21} + 1/2\lambda_3{}^2 x_{32} & \text{in } B_3 \\ (-1/2\lambda_1{}^2+\lambda_1\lambda_3)x_{32} + 1/2\lambda_1{}^2 x_{21} & \text{in } B_1 \\ -1/18 x_{13} & \text{in } B_0 \end{cases}$$

$$D\cdot\rho_{3x} = \begin{cases} (\lambda_1\lambda_2+5/2\lambda_3{}^2-\lambda_3)y_{12} + \lambda_2\lambda_3 y_{23} + \lambda_3\lambda_1 y_{31} & \text{in } B_3 \\ (-1/2\lambda_1{}^2+\lambda_1\lambda_2)y_{23} + 1/2\lambda_1{}^2 y_{31} & \text{in } B_1 \\ (-1/2\lambda_2{}^2+\lambda_2\lambda_1)y_{31} + 1/2\lambda_2{}^2 y_{23} & \text{in } B_2 \\ -1/18 y_{12} & \text{in } B_0 \end{cases}$$

$$D\cdot\rho_{3y} = \begin{cases} (\lambda_1\lambda_2+5/2\lambda_3{}^2-\lambda_3)x_{21} + \lambda_2\lambda_3 x_{32} + \lambda_3\lambda_1 x_{13} & \text{in } B_3 \\ (-1/2\lambda_1{}^2+\lambda_1\lambda_2)x_{32} + 1/2\lambda_1{}^2 x_{13} & \text{in } B_1 \\ (-1/2\lambda_2{}^2+\lambda_2\lambda_1)x_{13} + 1/2\lambda_2{}^2 x_{32} & \text{in } B_2 \\ -1/18 & \text{in } B_0. \end{cases}$$

Now let

$$(5.1.10)\quad M_{ji} = z_{ix}x_{ji} + z_{iy}y_{ji},\quad i,j = 1,\ 2,\ 3,\quad i\neq j.$$

These six quantities represent the directional derivatives at vertex i with respect to the direction vector (x_{ji},y_{ji}), which represents a directed edge of the triangle. It will be convenient to use the additional abreviations

$$(5.1.11)\quad Q_{ij} = 1/2(M_{ji} + M_{ij}),\quad C_{ij} = 1/2(M_{ji} - M_{ij}) - z_{ji}.$$

Note that for a linear function $z = f(x,y)$, $M_{ji} = -M_{ij} = z_{ji}$, and for a quadratic function, $M_{ji} - M_{ij} = 2z_{ji}$. As a consequence, the coefficients Q_{ij} and C_{ij} vanish for linear functions, and the coefficients C_{ij} for quadratic functions. We also denote by

$$(5.1.12)\quad L_i,\ i = 1,\ 2,\ 3,$$

the euclidean length of the edge opposite to vertex i. The Clough-Tocher element is now of the form:

$$\begin{aligned}(5.1.13)\quad z = {} & \lambda_1 z_1 + \lambda_2 z_2 + \lambda_3 z_3 \\ & + Q_{23}\lambda_2\lambda_3 + Q_{31}\lambda_3\lambda_1 + Q_{12}\lambda_1\lambda_2 \\ & + C_{23}V_1 + C_{31}V_2 + C_{12}V_3,\end{aligned}$$

where the functions V_i, $i = 1,2,3$, are given by

$$V_1 = \lambda_2\lambda_3(\lambda_2-\lambda_3) + [3(L_2+L_3)(L_2-L_3)/L_1^2]\rho_1 - \rho_2 + \rho_3$$

$$V_2 = \lambda_3\lambda_1(\lambda_3-\lambda_1) + [3(L_3+L_1)(L_3-L_1)/L_2^2]\rho_2 - \rho_3 + \rho_1$$

$$V_3 = \lambda_1\lambda_2(\lambda_1-\lambda_2) + [3(L_1+L_2)(L_1-L_2)/L_3^2]\rho_3 - \rho_1 + \rho_2.$$

Note that the expression

$$\lambda_1 z_1 + \lambda_2 z_2 + \lambda_3 z_3$$

represents the linear element, namely the plane passing through the elevations given at the vertices of the triangle. The portion

$$Q_{23}\lambda_2\lambda_3 + Q_{31}\lambda_3\lambda_1 + Q_{12}\lambda_1\lambda_2$$

is a quadratic function in the entire triangle. Note that quadratic functions satisfy the Linear Derivative Condition (5.1.?). The remaining portion is a piecewise cubic function with respect to the barycentric partition. This function also satisfies the Linear Derivative Condition as a result of the choice of correction functions ρ_i.

It is easy to derive various expressions for the gradient of the Clough-Tocher function. The gradient components are the partial derivatives with respect to x, y, and may be written as

$$\begin{aligned}(5.1.14)\quad D\cdot z_x &= z_1y_{23} + z_2y_{31} + z_3y_{12} \\ &\quad + Q_{23}(\lambda_3y_{31}+\lambda_2y_{12}) + Q_{31}(\lambda_1y_{12}+\lambda_3y_{23}) + Q_{12}(\lambda_2y_{23}+\lambda_1y_{31}) \\ &\quad + C_{23}D\cdot V_{1x} + C_{31}D\cdot V_{2x} + C_{12}D\cdot V_{3x}\end{aligned}$$

$$\begin{aligned}D\cdot z_y &= z_1x_{32} + z_2x_{13} + z_3x_{21} \\ &\quad + Q_{23}(\lambda_3x_{13}+\lambda_2x_{21}) + Q_{31}(\lambda_1x_{21}+\lambda_3x_{32}) + Q_{12}(\lambda_2x_{32}+\lambda_1x_{13}) \\ &\quad + C_{23}D\cdot V_{1y} + C_{31}D\cdot V_{2y} + C_{12}D\cdot V_{3y},\end{aligned}$$

where

$$\begin{aligned}D\cdot V_{1x} &= \lambda_3(2\lambda_2-\lambda_3)y_{31} - \lambda_2(2\lambda_3-\lambda_2)y_{12} \\ &\quad + [3(L_2+L_3)(L_2-L_3)/L_1^2]D\cdot\rho_{1x} - D\cdot\rho_{2x} + D\cdot\rho_{3x}\end{aligned}$$

$$\begin{aligned}D\cdot V_{2x} &= \lambda_1(2\lambda_3-\lambda_1)y_{12} - \lambda_3(2\lambda_1-\lambda_3)y_{23} \\ &\quad + [3(L_3+L_1)(L_3-L_1)/L_2^2]D\cdot\rho_{2x} - D\cdot\rho_{3x} + D\cdot\rho_{1x}\end{aligned}$$

$$\begin{aligned}D\cdot V_{3x} &= \lambda_2(2\lambda_1-\lambda_2)y_{23} - \lambda_1(2\lambda_2-\lambda_1)y_{31} \\ &\quad + [3(L_1+L_2)(L_1-L_2)/L_3^2]D\cdot\rho_{3x} - D\cdot\rho_{1x} + D\cdot\rho_{2x}\end{aligned}$$

$$D\cdot V_{1y} = \lambda_3(2\lambda_2-\lambda_3)x_{13} - \lambda_2(2\lambda_3-\lambda_2)x_{21}$$
$$+ [3(L_2+L_3)(L_2-L_3)/L_1^2]D\cdot\rho_{1y} - D\cdot\rho_{2y} + D\cdot\rho_{3y}$$

$$D\cdot V_{2y} = \lambda_1(2\lambda_3-\lambda_1)x_{21} - \lambda_3(2\lambda_1-\lambda_3)x_{32}$$
$$+ [3(L_3+L_1)(L_3-L_1)/L_2^2]D\cdot\rho_{2y} - D\cdot\rho_{3y} + D\cdot\rho_{1y}$$

$$D\cdot V_{3y} = \lambda_2(2\lambda_1-\lambda_2)x_{32} - \lambda_1(2\lambda_2-\lambda_1)x_{13}$$
$$+ [3(L_1+L_2)(L_1-L_2)/L_3^2]D\cdot\rho_{3y} - D\cdot\rho_{1y} + D\cdot\rho_{2y}.$$

A more transparent formula for the gradient can be derived. For $\lambda_i = 1$, i=1,2,3, that is, at the three corners of the triangle, $\rho_{ix} = \rho_{iy} = 0$. It thus follows from (5.1.14) that

$$D\cdot z_{1x} = z_1y_{23} + z_2y_{31} + z_3y_{12} + (Q_{12}+C_{12})y_{31} + (Q_{31}-C_{31})y_{12}$$

$$D\cdot z_{2x} = z_1y_{23} + z_2y_{31} + z_3y_{12} + (Q_{23}+C_{23})y_{12} + (Q_{12}-C_{12})y_{23}$$

$$D\cdot z_{3x} = z_1y_{23} + z_2y_{31} + z_3y_{12} + (Q_{31}+C_{31})y_{23} + (Q_{23}-C_{23})y_{31}$$

$$D\cdot z_{1y} = z_1x_{32} + z_2x_{13} + z_3x_{21} + (Q_{12}+C_{12})x_{13} + (Q_{31}-C_{31})x_{21}$$

$$D\cdot z_{2y} = z_1x_{32} + z_2x_{13} + z_3x_{21} + (Q_{23}+C_{23})x_{21} + (Q_{12}-C_{12})x_{32}$$

$$D\cdot z_{3y} = z_1x_{32} + z_2x_{13} + z_3x_{21} + (Q_{31}+C_{31})x_{32} + (Q_{23}-C_{23})x_{13},$$

where z_{ix}, z_{iy}, i=1,2,3, are the prescribed gradient components at the vertices of the triangle. It follows that

$$\lambda_1D\cdot z_{1x}+\lambda_2D\cdot z_{2x}+\lambda_3D\cdot z_{3x} = z_1y_{23}+z_2y_{31}+z_3y_{12}$$
$$+ Q_{23}(\lambda_3y_{31}+\lambda_2y_{12})+Q_{31}(\lambda_1y_{12}+\lambda_3y_{23})+Q_{12}(\lambda_2y_{23}+\lambda_1y_{31})$$
$$- C_{23}(\lambda_3y_{31}-\lambda_2y_{12})-C_{31}(\lambda_1y_{12}-\lambda_3y_{23})-C_{12}(\lambda_2y_{23}-\lambda_1y_{31})$$

$$\lambda_1 D\cdot z_{1y}+\lambda_2 D\cdot z_{2y}+\lambda_3 D\cdot z_{3y} = z_1x_{32}+z_2x_{13}+z_3x_{21}$$
$$+ Q_{23}(\lambda_3x_{13}+\lambda_2x_{21})+Q_{31}(\lambda_1x_{21}+\lambda_3x_{32})+Q_{12}(\lambda_2x_{32}+\lambda_1x_{13})$$
$$- C_{23}(\lambda_3x_{13}-\lambda_2x_{21})-C_{31}(\lambda_1x_{21}-\lambda_3x_{32})-C_{12}(\lambda_2x_{32}-\lambda_1x_{13}).$$

Formula (5.1.14) can now be rewritten as

$$(5.1.15)\quad z_x = \lambda_1 z_{1x} + \lambda_2 z_{2x} + \lambda_3 z_{3x} + C_{23}A_{1x} + C_{31}A_{2x} + C_{12}A_{3x}$$
$$z_y = \lambda_1 z_{1y} + \lambda_2 z_{2y} + \lambda_3 z_{3y} + C_{23}A_{1y} + C_{31}A_{2y} + C_{12}A_{3y},$$

where

$$A_{1x} = V_{1x} + (\lambda_3 y_{31}-\lambda_2 y_{12})/D, \quad A_{1y} = V_{1y} + (\lambda_3 x_{13}-\lambda_2 x_{21})/D$$

$$A_{2x} = V_{2x} + (\lambda_1 y_{12}-\lambda_3 y_{23})/D, \quad A_{2y} = V_{2y} + (\lambda_1 x_{21}-\lambda_3 x_{32})/D$$

$$A_{3x} = V_{3x} + (\lambda_2 y_{23}-\lambda_1 y_{31})/D, \quad A_{3y} = V_{3y} + (\lambda_2 x_{32}-\lambda_1 x_{13})/D.$$

To sum up, given three points with their elevations and gradients, that is, given 15 quantities

$$x_i,\ y_i,\ z_i,\ z_{ix},\ z_{iy},\quad i = 1,\ 2,\ 3,$$

the Clough-Tocher element defines a surface that assumes those prescribed elevations and derivatives (gradients). In order to calculate the elevation at an arbitrary specified point, we

- compute the auxiliary quantities Q_{ij}, C_{ij} and L_i from the above 15 quantities
- evaluate the correction functions ϱ_i for the barycentric coordinates λ_1, λ_2, λ_3 with respect to the three given points in the plane
- determine the values V_i and enter them into the Clough-Tocher formula (5.1.13). The gradient components are computed analogously using formula (5.1.15).

5.2 SURFACE GENERATION OVER A TRIANGULATED REGION

In order to pass a smooth surface through given elevations at irregularly spaced points, one may partition the region in which the surface is to be defined into triangles, specify gradients along with the evevations at the vertices of these triangles, and generate the resulting Clough-Tocher patch in every triangle. The Clough-Tocher patches are designed in such a fashion that they fit together smoothly along common boundary edges. However, there is still a missing link.

In order to fully define our synthetic surface, we have yet to provide the gradients (tangential planes) at the vertices of the triangulation. There are several approaches to finding suitable gradient values. One is to examine neighboring elevations and to estimate a suitable position of the tangential plane at the point in question by using local interpolation or least squares regression. The success of spline techniques suggests a different approach. A spline can be interpreted as an idealized mechanical structure consisting of "thin beams" which, when forced through specified points, assume a position in which a surrogate elastic energy is minimized. Since oscillatory behavior is associated with high elastic energy, minimizing elastic energy tends to minimize oscillations. In this work, we extend this approach to two dimensions. We consider a mechnical structure consisting of thin beams of equal "thickness" along the edges. They are joined together at vertices by small "thin plates" which represent tangential planes forced to be met by the adjacent thin beams. The sum of the surrogate energies of all thin beams is then minimized. This is achieved by varying the positions of the thin plates to which the adjacent thin beams must be tangential while passing through prescribed elevations and satisfying other side conditions that may have been specified for selected vertices. Once the idealized mechanical structure has found its optimal, that is, energy-minimal position, the gradient at each vertex is defined by the tilt of its thin plate. The resulting surface is supported by thin beams much as the fabric of an umbrella is spanned by its ribs.

Mathematically, the surrogate elastic energy is a positive definite quadratic form in those parameters that are permitted to vary. Setting the partial

derivatives of the energy with respect to these variables to zero yields the optimality conditions in the form of a large system of equations.

The classic Gauss-Seidel method for solving a system of linear equations is an iterative procedure where each iteration consists of passing in sequence through all the equations of the system. Each equation is used for determining a new value for a particular unknown variable while keeping the other unknowns fixed. The method thus requires an initial value for each unknown. To start, the first equation is transformed into a linear equation of only the first unknown by substituting into this equation the initial values of all remaining variables. This equation then yields an improved value for the first unknown variable. This value, along with initial values for the third and subsequent variables, enters the second equation, which then yields an equation for the second variable alone, and so on.

We modify the Gauss-Seidel procedure slightly. To this end we observe that each variable of the system of linear equations is "associated" with a particular vertex. The associated variables at each vertex satisfy "local optimality conditions". These optimality conditions have a structural interpretation: They express the conditions for the structural parameters represented by the associated variables to assume minimum energy values, supposing that the structural parameters at all other vertices remain fixed. The local optimality conditions again take the form of linear equations in the variables associated with the vertex. In a typical case, the thin plate at the vertex is tilted into the best position it can assume, given the tilts and elevations at neighboring vertices. The local optimality conditions then define this locally optimal tilt. It can be shown that:

(5.2.1) THEOREM: The linear system of equations for optimizing the position of the idealized mechanical structure is equivalent to the combination of all local optimality conditions.

Our variation of the Gauss-Seidel method now is to pass through all vertices in sequence, solving at each vertex the local optimality conditions. It was a

major concern at the start of this project whether this procedure was computationally feasible. We found that the time needed for a Gauss-Seidel iteration of the kind described above was well within an acceptable time frame. As discussed in Section 6, this marks a main accomplishment of this feasibility study.

In what follows, we will list the local optimality conditions for various sets of associated variables. We distinguish several "types of vertices" according to their kinds of associated variables. A particularly important case is the one in which the tangent to the contour curve is given along with its elevation. In this case, the direction of the gradient is given -- it is perpendicular to the contour tangent -- and the only variable to be determined is the length or, rather, a positive or negative "gradient multiplier" γ. It represents the only variable associated with a vertex of this type.

We use a three letter code to characterize vertex types. The first letter of the code refers to elevation: it is 'E' if the elevation is given, and 'N' otherwise. In the latter case, the elevation is an associated variable. The second and third letters refer to gradient components z_x and z_y, respectively. Letter 'X' in the second position indicates that the x-component of the gradient is given, and 'N' in this position indicates that it is not. Analogously, letter 'Y' in the third position indicates that the y-component of the gradient is given, and 'N' indicates that it is not. Finally, the combination 'RN' is found in positions two and three, if the gradient direction, but not the gradient itself, is specified. For example, 'ERN' signals the case in which both elevation and gradient direction are prescribed, leaving the gradient multiplier as the only associated variable. Any occurrence of 'N' indicates an unknown associated variable. In particular 'NNN' is used if all three quantities, elevation and gradient components, are to be determined.

The surrogate energy of a thin beam of length L is given by

$$(5.2.2) \qquad E = \int_{s=0}^{s=L} z''(s)^2 ds.$$

It is a well-known result of the theory of thin beams that the above expression is minimized by cubic functions of the distance s along the projection of the beams into the plane. These cubic functions are uniquely determined by the elevations and slopes at the end points ("Hermite interpolation"). The following local optimality conditions are derived using these facts.

In describing local optimality conditions at vertex i, we let the vertices j run through the "star" of vertex i, namely the following set of vertices j:

$$(5.2.3) \quad S(i) = \{ j : j \text{ is connected by an edge to vertex } i \}.$$

The abbreviations (5.1.10) and

$$(5.2.4) \quad L_{ji} = \sqrt{x_{ji}^2 + y_{ji}^2}$$

are used to denote the edge-directional derivatives and the distances between vertices, respectively. In addition, we will need the quantities

$$(5.2.5) \quad \mu_{ji} = \mu_{ix} x_{ji} + \mu_{iy} y_{ji},$$

where

$$\mu_{ix}, \mu_{iy}, \mu_{ix2} + \mu_{iy2} = 1$$

are the components of the gradient direction at vertex i normalized to length 1. The resulting optimality conditions for all types of vertices, except type 'EXY', are displayed in Figure 19. 'EXY' represents the fully specified case in which there are no associated variables to be determined.

To sum up, the vertices of the triangulation are divided prior to surface generation into types depending on which surface parameters are given and which have to be determined. These types are described by the letter codes

$$(5.2.6) \quad \begin{array}{l} \text{'ERN', 'ENN', 'EXN', 'ENY', 'EXY',} \\ \text{'NRN', 'NNN', 'NXN', 'NNY', 'NXY'.} \end{array}$$

For each of these ten types -- except the fully specified case 'EXY' -- there is an update formula by which the surface parameters at the triangulation vertices are calculated from the current values of the surface parameters at neighboring vertices. Sequentially updating every vertex in this fashion constitutes a Gauss-Seidel iteration. Such iterations are repeated until the changes in the variables remain within a given tolerance or a specified limit on the number of iterations is reached.

5.3 FINDING THE RIGHT TRIANGLE

Suppose a surface is specified in terms of triangular Clough-Tocher patches. In order to evaluate the elevation at a given point in that surface, a triangle containing that point needs to be found. In what follows we describe a method for finding such a triangle. This method is intended for applications in which not just one point, but sequences of such points are given, most of which are moreover close to each other. The sets of sequential contour points as they arise from digitized contour information are a case in point. A closely spaced regular grid is another. In these cases, a given point in the sequence will lie with high probability in the same triangle or in a triangle directly adjacent to the triangle of the previous point.

The method we use for finding a triangle containing a given point relies for its efficiency on the above observation. Before searching for the triangle of a given point, the method requires that an arbitrary starting point be specified for which a triangle containing it is known. We then move from this starting point straight towards the given point until we reach the boundary of the starting triangle or the given point itself, whichever happens first. If the boundary is reached at a non-vertex point, that is, somewhere in the interior of a boundary edge, then the unique adjacent triangle can be readily identified using our triangulation data structure, and we continue moving in that triangle as far as possible or necessary. In the unlikely case that a vertex is encountered, all triangles adjacent to this vertex are examined in sequence until one is found in which progress can be made towards the given point. The processes are repeated until the given point is reached. The last triangle in

Case ERN: elevation z_i and gradient direction (μ_{iz}, μ_{iy}) specified, gradient multiplier γ_i to be determined.

$$\gamma_i \times 2\sum_j \frac{(\mu_{ji})^2}{(L_{ji})^3} = \sum_j \frac{\mu_{ji}}{(L_{ji})^3}[3z_{ji} + M_{ij}]$$

Case EXN: elevation z_i and gradient component z_{iz} specified, gradient component z_{iy} to be determined.

$$z_{iy} \times 2\sum_j \frac{(y_{ji})^2}{(L_{ji})^3} = \sum_j \frac{y_{ji}}{(L_{ji})^3}[3z_{ji} - 2x_{ji}z_{iz} + M_{ij}]$$

Case ENY: elevation z_i and gradient component z_{iy} specified, gradient component z_{iz} to be determined.

$$z_{iz} \times 2\sum_j \frac{(x_{ji})^2}{(L_{ji})^3} = \sum_j \frac{x_{ji}}{(L_{ji})^3}[3z_{ji} - 2y_{ji}z_{iy} + M_{ij}]$$

Case ENN: elevation z_i specified, gradient (z_{iz}, z_{iy}) to be determined.

$$z_{iz} \times 2\sum_j \frac{(x_{ji})^2}{(L_{ji})^3} + z_{iy} \times 2\sum_j \frac{x_{ji}y_{ji}}{(L_{ji})^3} = \sum_j \frac{x_{ji}}{(L_{ji})^3}[3z_{ji} + M_{ij}]$$

$$z_{iz} \times 2\sum_j \frac{x_{ji}y_{ji}}{(L_{ji})^3} + z_{iy} \times 2\sum_j \frac{(y_{ji})^2}{(L_{ji})^3} = \sum_j \frac{y_{ji}}{(L_{ji})^3}[3z_{ji} + M_{ij}]$$

Case NRN: gradient direction (μ_{iz}, μ_{iy}) specified, elevation z_i and gradient multiplier γ_i to be determined.

$$z_i \times 6\sum_j \frac{1}{(L_{ji})^3} + \gamma_i \times 3\sum_j \frac{\mu_{ji}}{(L_{ji})^3} = \sum_j \frac{1}{(L_{ji})^3}[6z_j + 3M_{ij}]$$

$$z_i \times 3\sum_j \frac{\mu_{ji}}{(L_{ji})^3} + \gamma_i \times 2\sum_j \frac{(\mu_{ji})^2}{(L_{ji})^3} = \sum_j \frac{(\mu_{ji}}{(L_{ji})^3}[3z_j + M_{ji}]$$

Figure 19. Optimality Conditions for Vertices by Type.

Case NXN: gradient component z_{ix} specified, elevation z_i and gradient component z_{iy} to be determined.

$$z_i \times 6\sum_j \frac{1}{(L_{ji})^3} + z_{iy} \times 3\sum_j \frac{y_{ji}}{(L_{ji})^3} = \sum_j \frac{1}{(L_{ji})^3}[6z_j - 3x_{ji}z_{ix} + 3M_{ij}]$$

$$z_i \times 3\sum_j \frac{y_{ji}}{(L_{ji})^3} + z_{iy} \times 2\sum_j \frac{(y_{ji})^2}{(L_{ji})^3} = \sum_j \frac{y_{ji}}{(L_{ji})^3}[3z_j - 2x_{ji}z_{ix} + M_{ij}]$$

Case NNY: gradient component z_{iy} specified, elevation z_i and gradient component z_{ix} to be determined.

$$z_i \times 6\sum_j \frac{1}{(L_{ji})^3} + z_{ix} \times 3\sum_j \frac{x_{ji}}{(L_{ji})^3} = \sum_j \frac{1}{(L_{ji})^3}[6z_j - 3y_{ji}z_{iy} + 3M_{ij}]$$

$$z_i \times 3\sum_j \frac{x_{ji}}{(L_{ji})^3} + z_{ix} \times 2\sum_j \frac{(x_{ji})^2}{(L_{ji})^3} = \sum_j \frac{x_{ji}}{(L_{ji})^3}[3z_j - 2y_{ji}z_{ix} + M_{ij}]$$

Case NNN: no parameters specified, elevation z_i and gradient (z_{ix}, z_{iy}) to be determined.

$$z_i \times 6\sum_j \frac{1}{(L_{ji})^3} + z_{ix} \times 3\sum_j \frac{x_{ji}}{(L_{ji})^3} + z_{iy} \times 3\sum_j \frac{y_{ji}}{(L_{ji})^3} = \sum_j \frac{1}{(L_{ji})^3}[6z_j + 3M_{ij}]$$

$$z_i \times 3\sum_j \frac{x_{ji}}{(L_{ji})^3} + z_{ix} \times 2\sum_j \frac{(x_{ji})^2}{(L_{ji})^3} + z_{iy} \times 2\sum_j \frac{x_{ji}y_{ji}}{(L_{ji})^3} = \sum_j \frac{x_{ji}}{(L_{ji})^3}[3z_j + M_{ij}]$$

$$z_i \times 3\sum_j \frac{y_{ji}}{(L_{ji})^3} + z_{ix} \times 2\sum_j \frac{x_{ji}y_{ji}}{(L_{ji})^3} + z_{iy} \times 2\sum_j \frac{(y_{ji})^2}{(L_{ji})^3} = \sum_j \frac{y_{ji}}{(L_{ji})^3}[3z_j + M_{ij}]$$

Case NXY: gradient (z_{ix}, z_{iy}) specified, elevation z_i to be determined.

$$z_i \times 2\sum_j \frac{1}{(L_{ji})^3} = \sum_j \frac{1}{(L_{ji})^3}[2z_j - M_{ji} + M_{ij}]$$

Figure 19. (Continued)

the chain of triangles obtained in the course of this procedure will contain the given point (Figure 20).

It is clear that this procedure will work best for a sequence of points in which the previous point can serve as a close starting point for the task of locating the given point in a triangle. For points in a regular grid arranged by sequential rows the following procedure is used. Locate the first point of the first row from an arbitrary starting point. Make a note of the first row and its triangle for further reference. Use it also as starting point for the second point in the first row. Then use the second point and its triangle as starters for locating the third point, and so on, until the end of the row is reached. Then retrieve the first point and its triangle and use them as starters for locating the first point in the second row. This point and its triangle are again kept for further reference, while the second row is traversed in the same manner as the first row. This process is repeated for the remaining rows.

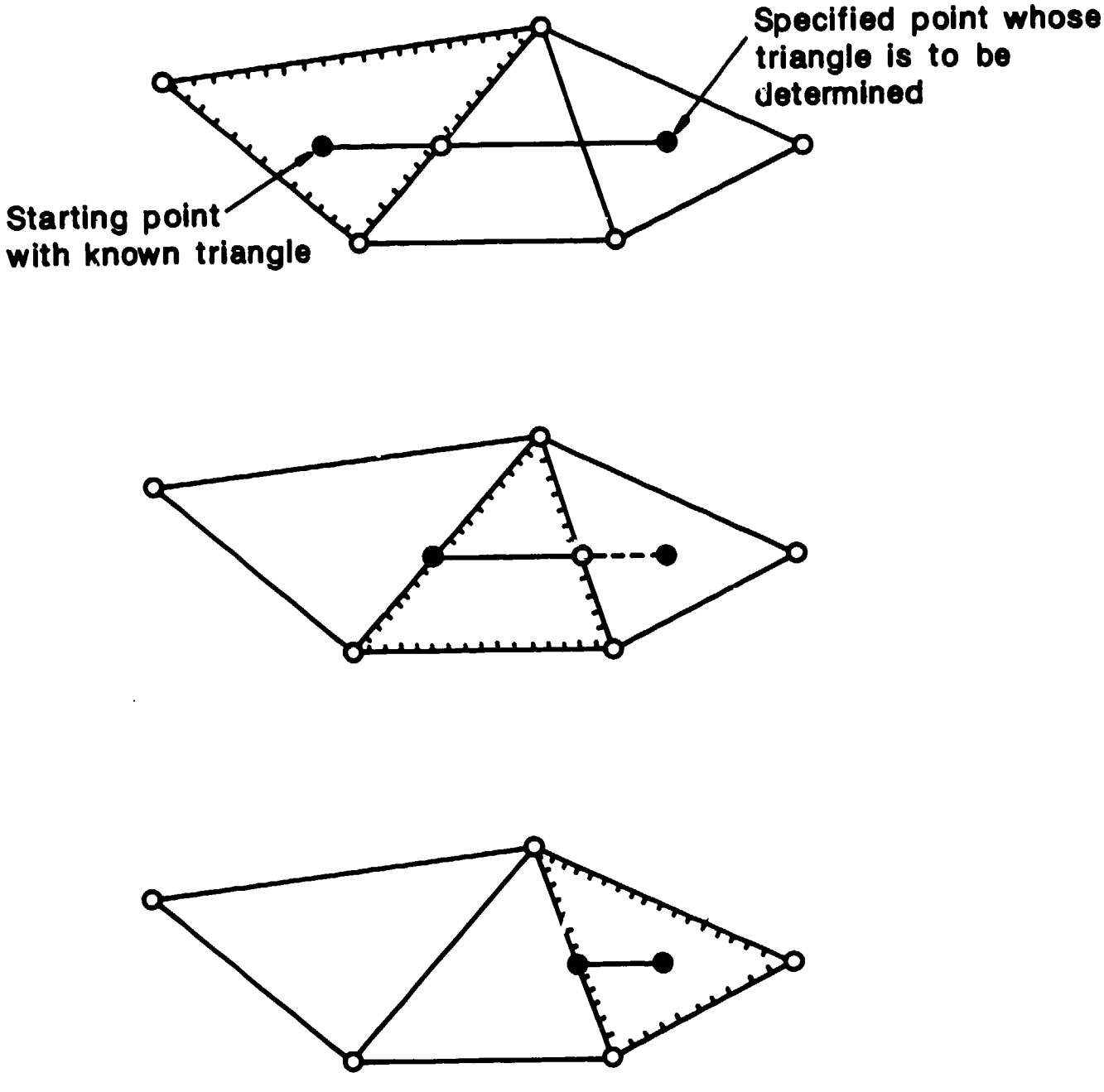

Figure 20. Line Search for a Triangle Containing a Specified Point.

6. RESULTS AND CONCLUSIONS

In order to test the computational feasibility of the approach proposed in the previous sections, a pilot implementation was applied to the Mustang Mountain data set (see Section 2). CPU times were recorded and experience about memory requirements was gained. As a preliminary testing procedure the residuals obtained when trying to recover the contour elevations were gathered and analyzed. After describing the set-up of the experiment, we report its results and the observed computational effort.

6.1 SETTING UP THE EXPERIMENT

The pilot implementation consists of several independent modules whose output files serve as input files to subsequent modules. The interrelationship of these modules and their interconnecting files is schematically described in Figure 21.

The original input file has the format (see Section 2) of Digital Graphic Recorder Data (DGR) and refers in our case to the Mustang Mountain, Fort Huachuca, area. These data are input into the module EDIT, which extracts and edits contour information as described in Section 2 of this report. The resulting edited digital contour file is then sampled and contour tangents are determined in module THIN. The sampling method is described in Section 3 of this report. The resulting sample is first fed to the module VORONOI (see Section 4), which determines a Voronoi triangulation of the sample points. Accordingly, the main output of this module is a "triangle table" that lists the vertices and neighbors of each triangle. In addition, duplicate sample points are identified and points that need to be added, such as map corners, are recorded. This information is utilized by the UPDATE module which creates the final "data base". This data base is needed along with the triangle table to generate the surface in module SURFACE. This module also provides options for evaluating the elevation of either random points or points in a regular

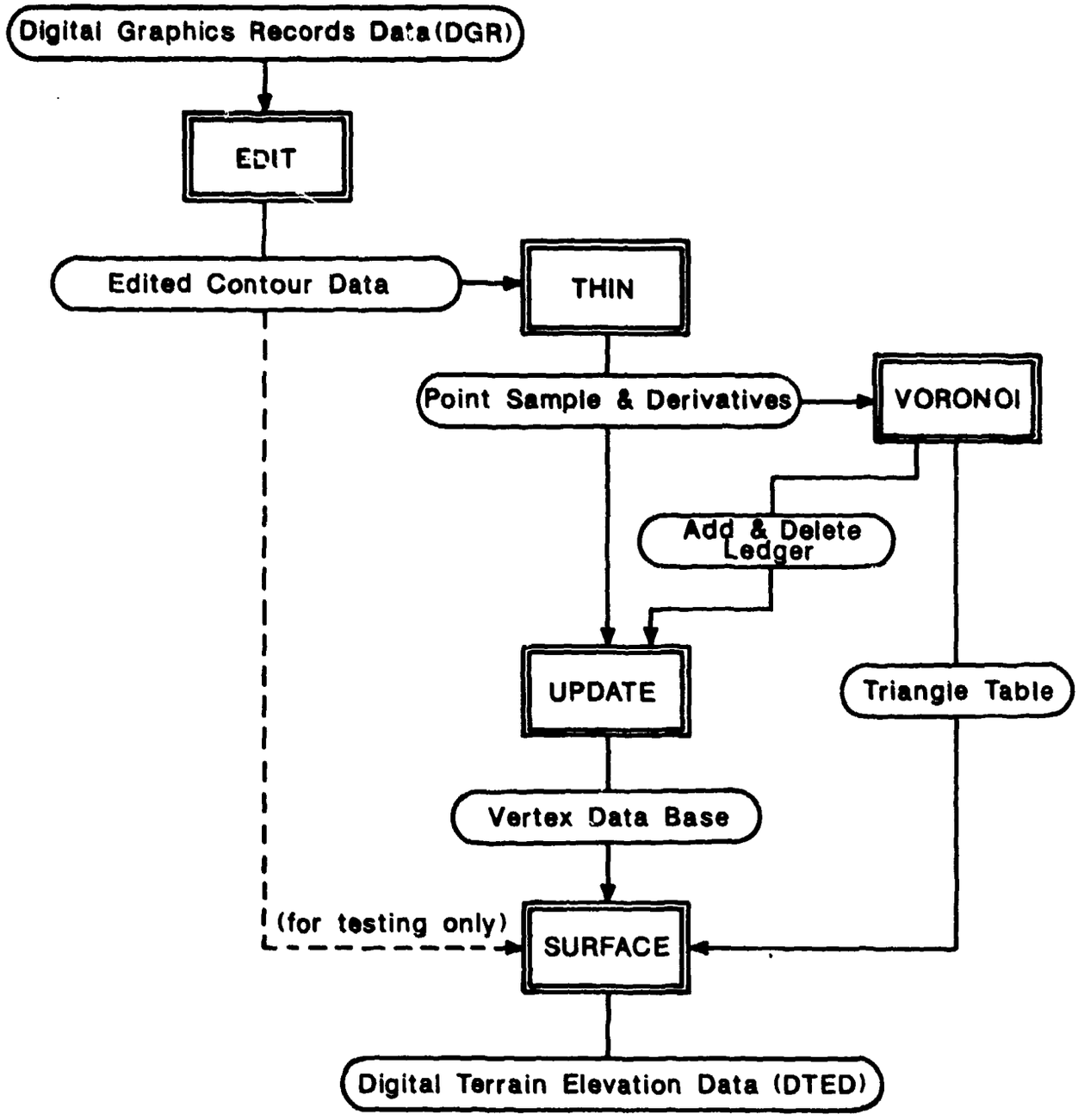

Figure 21. System Layout.

rectangular grid. The design of the testing procedure is based on the following observation. When the sample points enter the triangulation process they are considered as points in a plane. No longer do they hold any direct relationship to other points in the contour as such. Therefore, these original contour points are in essence independent of the sampled ones. The testing procedure consists of evaluating the surface at the original contour points and comparing the results to the given elevations, yielding a set of residuals. These residuals provide an indication of the ability of our algorithm to "recover" the original data.

The full original data set contains features such as lake shores, lake hatchings, dams, and peak elevations, which the algorithm in its present state of development does not yet handle in an accurate fashion. Indeed, the primary goal of the effort reported here is to establish computational feasibility. Also some limitations of the testing procedure itself need to be pointed out.

First, the digitized data themselves carry a "digitization error" so that it is not necessarily desirable to reproduce the digitized data precisely. Indeed, if the recovered contours represent a "smoothing" of the digitized lines, they may be more representative of the true surface than the given digitized contour points.

Second, in flat terrain, the nonsampled digitized contour points tend to be close to the boundary of triangles, so that the behavior of the surface in their interior is monitored to a less extent than in steep terrain, where more contour lines cut through the interior of triangles. In order to retain interior monitoring, the sample was deliberately chosen somewhat smaller than indicated for the purpose of improved accuracy, but still large enough to test computational feasibility.

Third, the vertical deviations measured by the residuals tend to be disproportionally large in steep terrain. Ideally, the 3-dimensional distances from the true data points to the generated surface should be evaluated in order to determine the accuracy of the latter. In flat terrain the vertical

deviation, that is, the residual, provides a good approximation to the true distance. However, this is not the case for steep terrain. For this reason we introduce an

$$(6.1.1)\quad \text{"adjusted residual"} = \frac{\text{residual}}{\sqrt{m^2 + 1}}$$

based on a measure m of "steepness" of terrain. The length of the gradient (5.1.16) appears to be a natural measure of steepness. However, the example of a mountain peak, where the slope is by definition zero, shows that the slope at a single point is not a good indicator of steepness. For the purposes of this report, we determine the triangle containing the point in question and then chose the vertex slope of largest magnitude, taking into account that the unit length in the plane is 20 feet (Section 2.1). The adjusted residual would represent the 3-dimensional surface precisely if the surface were linear, that is, a tilted plane in a suitable neighborhood of the data point. In general, however, it is still an approximation, but a better one than the unadjusted residual. For horizontal terrain, $m = 0$ and the adjusted residual equals the original one. In all other cases, the adjusted residual is smaller. In our experiment, we collected statistics on both types of residuals.

In Mandel, Witzgall, and Bernal (86), the results of a first test run were reported. For this run, the unadjusted residuals for the full set of digitized contour points were collected, including lake hatchings and dams, even though the algorithm is not yet equipped to handle nonsmooth terrain, as pointed out above. Nevertheless, 95% of the residuals were between ± 12.5 feet, indicating that at least 90% did not deviate more than halfway to the next contour line. Furthermore, an analysis of the biggest residuals led to the discovery of several contour lines whose altitudes had been apparently miscoded. For the purpose of the more extensive experiments reported here, the original data set was purged of lake hatchings and the altitudes were recoded for the above contour lines. In addition, one contour line representing a dam was removed from consideration. In what follows, the results are based on this "sanitized" data set.

6.2 RESULTS

The experiment reported here comprises three runs, all for the same "sanitized" Mustang mountain data set of about 38,000 sample points giving rise to about 75,000 triangles. For the first and main run, a histogram of the (unadjusted) residuals is displayed in Figure 22. In addition, the standard statistical quantities such as expected value (=average), standard deviation, maximum and average absolute value are reported, the latter two also for the adjusted residuals (6.1.1). The second run did not utilize the contour tangent information and the third run used linear rather than Clough-Tocher interpolation on the given set of triangles.

The timing of the procedure is broken down by major steps, including the generation of a 901 X 901 rectangular grid, which we expect to be of a size relevant to prospective applications. The calculations were timed and carried out on a VAX 11/780 system at the Engineer Topographic Laboratories. The observed CPU times for the different steps follow

- o Step 1: Editing Digitized Input Data ..... 25 min
- o Step 2: Thining and Tangent Determination .... 9 min
- o Step 3: Voronoi Triangulation and Update ..... 21 min
- o Step 4: Surface Generation ..... 20 min
- o Step 5: Grid Determination (901 X 901) ..... 26 min

  Total CPU Time ..... 101 min

The timing of Step 5 represents an improvement over the one reported in Mandel, Witzgall, and Bernal (86).

As discussed before, the testing procedure consisted of calculating and analysing the residuals of the elevation of the original contour points. The histogram of these residuals is displayed in Figure 22. Other statistics calculated include:

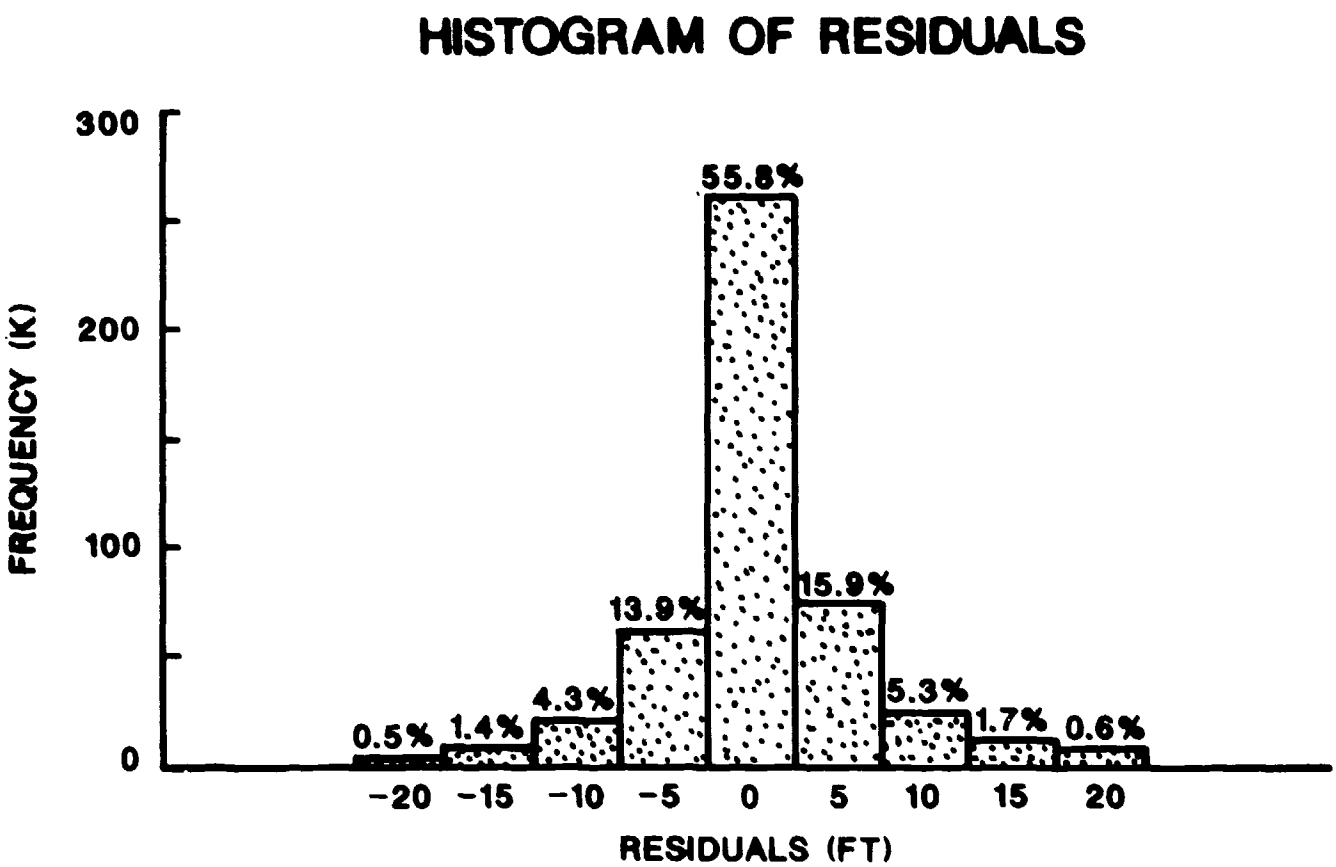

Figure 22. Histogram of Residuals.

- o Expected Value (average residual) ..... 0.294 FT
- o Standard Deviation ..... 5.934 FT

- o Average Absolute Error ..... 3.595 FT
- o Maximum Absolute Error ..... 99.610 FT
- o Percent Absolute Values < 12.5 FT ..... 95.141 %
- o Number Absolute Values > 99.5 FT 1
- o Number of
  - Positive Residuals (overestimates) ... 222 417
  - Negative Residuals(underestimates) ... 205 365
  - Zero Residuals 38 678

- o Maximum Absolute Adjusted Residual ..... 65.588 FT
- o Average Absolute Adjusted Residual ..... 3.13_ FT
- o Percent Absolute Adjusted Values > 40 FT .... 0.020 %

The adjusted residuals are seen to be much smaller than the unadjusted ones. This indicates that big values of the (unadjusted) residuals are largely confined to steep terrain. Results for the second run are displayed below. It is seen that not to use tangential information results in a definite deterioration of accuracy:

- o Expected Value (average residual) ..... 0.209 FT
- o Standard Deviation ..... 6.034 FT

- o Average Absolute Error ..... 3.657 FT
- o Maximum Absolute Error ..... 160.619 FT
- o Percent Absolute Values < 12.5 FT ..... 95.044 %
- o Number Absolute Values > 99.5 FT 15
- o Number of
  - Positive Residuals (overestimates) ... 221 420
  - Negative Residuals(underestimates) ... 206 419
  - Zero Residuals 38 621

As expected, the third run, featuring linear interpolation, did not achieve the accuracy of the first run. It comes as somewhat of a surprise, however, that the loss of accuracy is not more pronounced. An analysis of the results shows that this is due to the extraordinary large number of zero residuals. This phenomenon, in turn, is an artefact of the digitization: digitized contour lines contain many groups of successive points that lie on a common line. If two of such points are vertices of the same triangle, then they and all intermediate points reproduce elevation under linear interpolation. As we pointed out earlier, the precise reproduction of digitized points is not necessarily desirable because the latter carry digitization errror. Below are the statistics for the linear run:

- o Expected Value (average residual) ..... 0.326 FT
- o Standard Deviation ..... 5.856 FT

- o Average Absolute Error ..... 3.557 FT
- o Maximum Absolute Error ..... 106.557 FT
- o Percent Absolute Values < 12.5 FT ..... 95.277 %
- o Number Absolute Values > 99.5 FT 4
- o Number of
  - Positive Residuals (overestimates) ... 190 778
  - Negative Residuals(underestimates) ... 175 542
  - Zero Residuals 100 140

6.3 CONCLUSIONS

Two particular concerns were our capability of obtaining the Voronoi triangulation of a sufficiently large set of sample points and of solving the large linear system for the gradients at the sample points, using commonly available computer resources. With respect to these concerns we found that samples of up to 70,000 points could be triangulated within a reasonable time frame. In fact, CPU time has been less of a problem than memory space, a limitation that was overcome by developing a decomposition method. The

Gauss-Seidel method described in Section 5.2 was found to be faster than anticipated, both with respect to the CPU time required by a single iteration (approximately 150 CPU seconds/iteration) and the speed of convergence (ten iterations). Alternative methods will perform as well or faster.

To sum up, we conclude that the computational effort of using nonlinear techniques for the generation of a smooth synthetic surface and subsequent regular grid is substantial, but within the bounds for routine calculations on a computer of medium size such as the VAX 11/780. Thus we were succesful in achieving the major goals of this feasibility study.

The residuals obtained when trying to recover the original digitized contour points are by and large comparable to the resolution of the digitized data. However, there are instances of very large residuals which may well be unacceptable for subsequent applications. Such discrepancies were expected because our method at this point still lacks the capability to handle cartographic features at which the actual terrain surface is not smooth, that is, it exhibits discontinuities of slope. Such features include lake shores, river banks, as well as some ridge and drainage lines. Modeling by a smooth surface may not be sufficiently accurate at such locations.

In order to achieve the full accuracy of our approach, the following measures need to be taken:

- o Cancel the smoothness requirements along lake shores, river banks, as well as along certain ridge and drainage lines.
- o Smooth digitized contour lines and determine tangent directions prior to thinning.
- o Investigate adaptive tolerance selection schemes for higher density sampling in rough terrain.
- o Compare local and global methods for specifying tangent planes.

REFERENCES

Bentley, J.L., Weide B.W. and A.C. Yao, 1980, "Optimal Expected Time Algorithms for Closest Point Problems", ACM Transactions on Mathematical Software, Vol. 6, No. 4, pp. 563-580.

Birkhoff, G. and L. Mansfield, 1974, "Compatible Triangular Finite Elements", Journal of Mathematical Analysis and Applications, Vol. 47, pp. 531-553.

Bowyer, A., 1981, "Computing Dirichlet Tesselations", The Computer Journal, Vol. 24, No. 2, pp. 162-166.

Clough, R.W. and J.L. Tocher, 1965, "Finite Element Stiffness Matrices for Analysis of Plates in Bending", Proceedings of the Conference on Matrix Methods in Structural Mechanics, Air Force Institute of Technology, Wright-Patterson A.F.B., Ohio.

Davis, D.M., Downing, J.A., and S. Zoraster, Algorithms for Digital Terrain Data Modeling, USAETL Technical Report ETL-0302, July 1982.

Defense Mapping Agency Hydrographic/Topographic Center, Informal Communications, 1985.

Douglas, D.H. and T.K. Poiker, 1973, "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Character", The Canadian Cartographer, Vol. 10, pp. 112-123.

Grotzinger, S., Danielson, B., Caldwell, D., and B. Mandel, 1984, The Contour-to-Grid Problem. An Evaluation of Simplicially Based Local Surface Estimators (SWIRL, CONTOCES, and TRENDVOL), Interim Paper, U.S. Army Engineer Topographic Laboratories, Fort Belvoir, Virginia.

Lang, T., 1969, "Rules for the Robot Draughtsmen", The Geographical Magazine, Vol. 42, pp. 50-51.

Lawson, C.L., 1972, Generation of a Triangular Grid with Application to Contour Plotting, Technical Memorandum No. 299, Jet Propulsion Laboratory, Pasadena, California.

Lawson, C.L., 1976, C^1-Compatible Interpolation Over a Triangle, Technical Memorandum No. 33-770, Jet Propulsion Laboratory, Pasadena, California.

Lawson, C.L., 1977, "Software for C^1 Surface Interpolation", in Mathematical Software III, J.R. Rice, editor, Academic Press, New York.

Mandel, B., Witzgall, C. and J. Bernal, 1986, "Non-Linear Contour-to-Grid Digital interpolation", to appear, Proceedings of the M,C&G-Conference, Oktober 1986.

Reumann, K. and A. Witkam, 1974, "Optimizing Curve Segmentation in Computer Graphics", International Computing Symposium, Amsterdam, Holland, pp. 467-472.

Peucker (Poiker), T.K. and D.H. Douglas, 1975, "Detection of Surface-Specific Points by Local Parallel Processing of Discrete Terrain Elevation Data", Comput. Graphics Image Process. Vol. 4,pp. 375-387.

Williams, C.M., 1978, "An Efficient Algorithm for the Piecewise Approximation of Planar Data", Computer Graphics and Image Processing, Vol. 8, pp. 286-293.

Williams, C.M., 1981, "Bounded Straight-Line Approximation of Digitized Planar Curves and Lines", Computer Graphics and Image Processing, Vol. 16, pp. 370-381.

Witzgall, C., Bernal, J. and B. Mandel, 1985, On Sampling and Triangulating Large Digitized Contour Data Sets, Interim Paper, U.S. Army Engineer Topographic Laboratories, Fort Belvoir, Virginia.

Zienkiewicz, O.C., 1971, The Finite Element Method in Structural and Continuum Mechanics, 2nd edition, McGraw Hill, New York.

Zoraster, S., Davis, D. and M. Hugus, 1984, Manual and Automated Line Generalization and Feature Displacement, Final Report for Contract DAAK70-82-C-0149, U.S. Army Engineer Topographic Laboratories, Fort Belvoir,Virginia, Report No. ETL-0359, 121 Pages.